DEPARTMENT OF THE ARMY
US ARMY RESEARCH, DEVELOPMENT, AND ENGINEERING COMMAND
AVIATION AND MISSILE RESEARCH DEVELOPMENT AND ENGINEERING CENTER
REDSTONE ARSENAL, AL 35898

"Database Translator (DATALATOR) for Integrated Exploitation"

10/31/2010

Phase I Final Technical Report

Reporting Period: 02/23/2010 – 10/31/2010

Principal Investigator: Len Yabloko

Sponsored by:

Defense Advanced Research Projects Agency (DOD)
Small Business Research Weapons Sciences Directorate AMRDEC

ARPA Order AY86-00

Issued by U.S. Army Aviation and Missile Command Under

Contract No. W31P4Q-10-C-0158

Effective Date of Contract:	02/23/2010
Contract Expiration Date:	10/31/2010

Next Generation Software
52 Strawtown Road, New City, NY 10956, 1(845) 633–3766, ngs@ontospace.net

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 10/31/2010	**2. REPORT TYPE** Phase I Final Technical Report	**3. DATES COVERED** *(From - To)* 02/23/2010 – 10/31/2010

4. TITLE AND SUBTITLE Database Translator (DATALATOR) for Integrated Exploitation Final Technical Report	**5a. CONTRACT NUMBER** W31P4Q-10-C-0158
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Len Yabloko	**5d. PROJECT NUMBER** PAN RTW 0G-10
	5e. TASK NUMBER A002
	5f. WORK UNIT NUMBER 0001AB
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NEXT GENERATION SOFTWARE 52 STRAWTOWN RD NEW CITY, NY 10956-6853	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency ATTN: AEO (Dr. Todd C. Hughes, Program Manager) 3701 North Fairfax Drive Arlington, VA 22203-1714	**10. SPONSOR/MONITOR'S ACRONYM(S)** DOD AMRDEC
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Reported are results of developing and testing the DATALATOR experimental prototype (TRL 4) designed to demonstrate its core functions based on Next Generation Software technology. The focus is on demonstrating tangible means for data users to be directly involved in meeting their operational objectives. The prototype employs innovative method of data integration that automates discovery of semantic model representing intended use of data. The automation is achieved by generating a graphical view (tree graph) of possible data flows based on a user initial indication of the flow target. Every path originating at the tree root represents a plausible chain of events and helps the user to identify the intended use of data. Once identified the data flow serves as a context for formulating and executing subsequent user-defined queries against the data source. This report contains details of technical implementation and empirical evaluation of the prototype during Phase I.

15. SUBJECT TERMS
Knowledge and decision bases; Intelligent knowledge querying; Semantic information; Ontology-based reasoning

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Len Yabloko
U	U	U	UU	50	**19b. TELEPHONE NUMBER** *(Include area code)* 845-633-3766

CONTENTS

List of Figures

SUMMARY

During the reported period the performer had advanced a previous TRL3 prototype OntoBase to TRL4 and conducted its empirical evaluation in accordance with phase I proposal (D092-007-0197). While the evaluation had shown the prototype had limited ability to scale up with an increasing volume of data, it had successfully demonstrated that the proposed method of semantic data integration fully addresses the main challenges put forth by the phase I solicitation: 1) highly automated mapping, and 2) bounded mapping effort. The evaluation shows in a rigorously empirical and quantitative fashion that the number of user interactions required for connecting a relational data source, discovering its intended use of data, and posing a simple query based on the intended semantics- is bounded to a relatively narrow range, even when using data sources from a wide range of domains, sizes and structure. The complexity of interactions as measured by number of required steps, also tend to stay within a reasonable range. A counterbalance measures to 1) and 2) has also been studied: 3) bounded answer scope, and 4) bounded answer quality. Even though no rigorous measurement was attempted, the answers to queries obtained in a specially crafted sets of experiments seemed to be reasonably suitable for DATALATOR to act as a "savant" guided by the user. Finally the prototype performance was studied using the data and benchmarks obtained from a recent study of analogous experimental prototype conducted at Oregon University. A measured time from completing the last step of the query formulation by user to obtaining corresponding answers increases exponentially to a volume of data in both studies. However, the increase measured in the experiments reported here is occurring earlier and slower than the benchmark. The report provides analysis of the prototype performance that traces these characteristics to Protégé platform on which OntoBase was originally developed. While TLR3 prototype was based on Protégé 3.3.1, the TRL4 prototype was migrated to Protégé 3.4.4 during phase I in attempt to mitigate the performance limitations. The experiments were then repeated, but did not show any significant improvement. Based on the finding in this report, and if the identified performance issues cannot be resolved, Next Generation Software will move OntoBase to an alternative platform that will mitigate the performance issues during the development of a commercial product prototype (TRL5) in phase II.

1 INTRODUCTION

1.1 Objective

As stated in the phase I solicitation DATALATOR is envisioned as a standalone appliance that interfaces with multiple data sources and performs three basic functions:

a) Source Modeling that would identify data formats and extract semantic models from data sources;
b) Alignment and Mapping that would establish semantic correspondence between data entities and relations from different sources;
c) Data Transformation that would translate multiple source data into integrated information space and allow execution of queries across the sources.

The focus of the DATALATOR research is structured text data sources (e.g., relational databases, XML databases, RDF/OWL databases, and spreadsheets).

Source: Todd Hughes, DARPA/IXO, Workshop on Informotion Integrotion, 2006

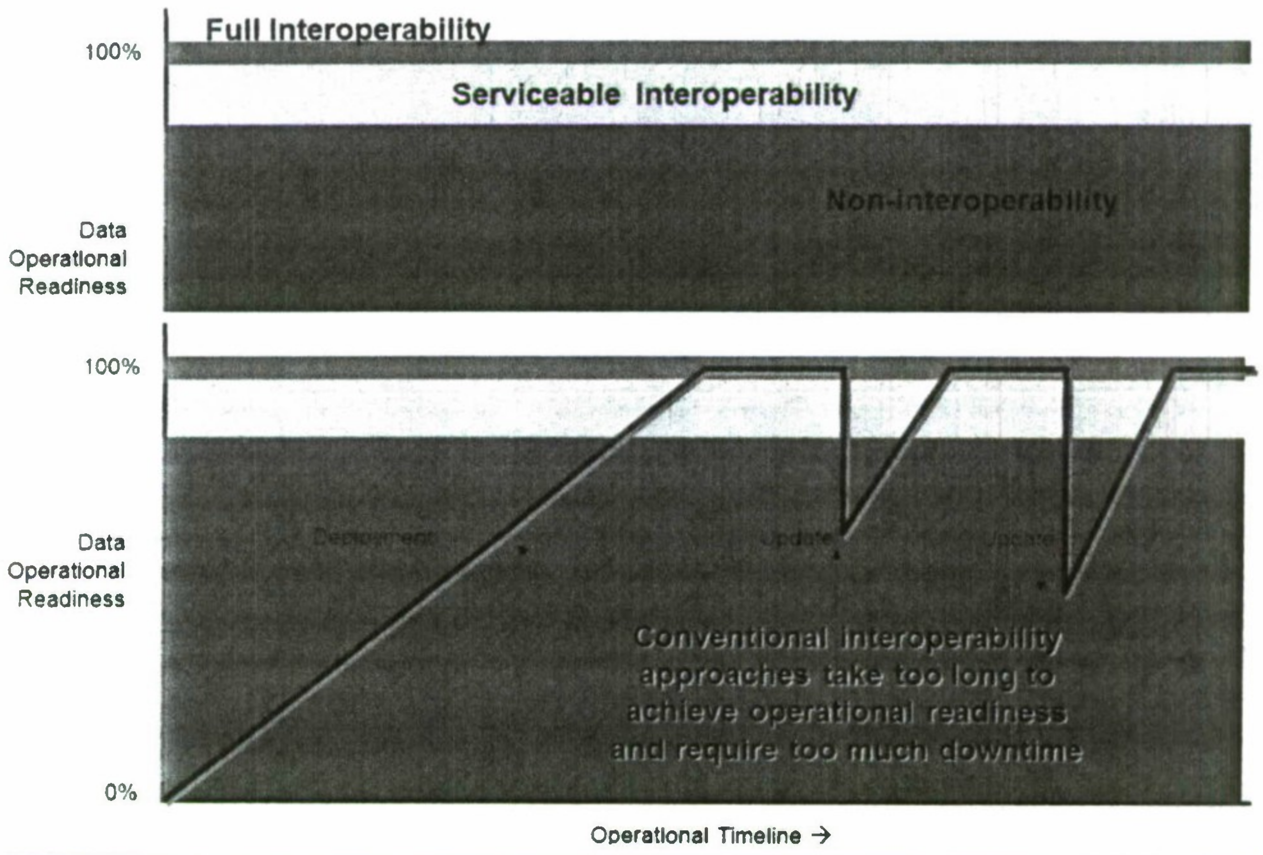


Figure 1: Conventional data interoperability

1.2 Current Approaches and Technical Barriers

Currently, there are several candidates for technology to rapidly aggregate and organize data from multiple sources [1]. Each technology is designed to operate under specific parameters and to produce results with specific characteristics. The operating parameters include: number of data sources, types of data, volume of data, its availability and volatility, as well as maintenance parameters: ready time and cost of operation, type of questions asked, interactive capabilities, required end user training. The resulting characteristics include: precision and speed of answers, robustness in response to changes, and flexibility in handling different types of questions.

The conventional data interoperability approaches can handle very large volumes of data and provide high precision and speed of answers, but take too long to achieve operational readiness and require too much downtime for maintenance (Figure 1). On the other hand, Semantic Web technologies offer

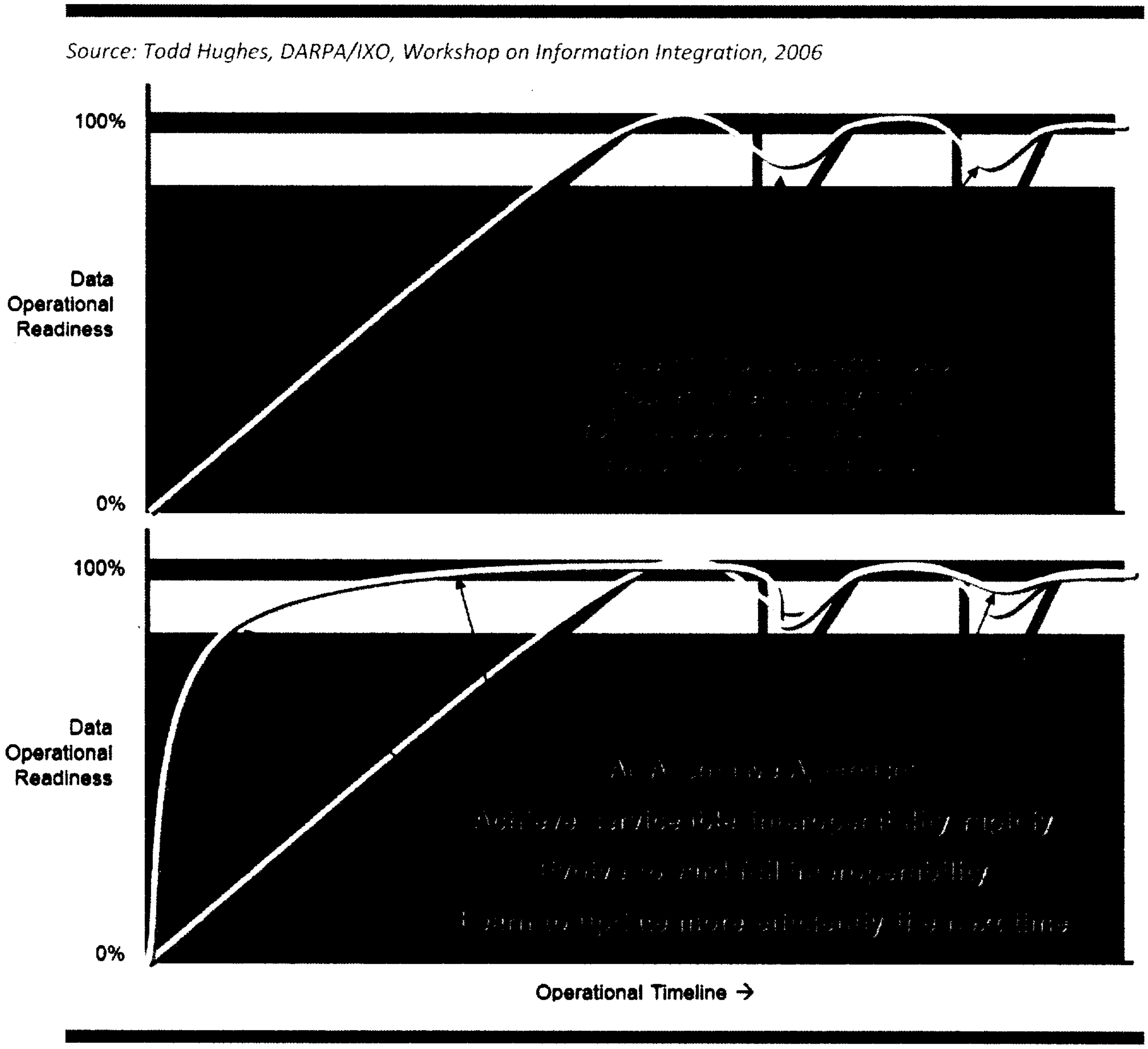


Figure 2: Alternatives to conventional approaches

greater robustness at the expense of lower data compression and throughput, but still suffer from enormous startup costs (Figure 2 top). At the same time, neither approach offers any tangible means for data users to be directly involved in meeting their operational objectives. Both conventional and Semantic Web technologies rely on highly trained personnel to anticipate user needs and translate it into semantic models which serve as a user interfaces. This disconnect is arguably the root cause of high cost in both cases. But more importantly, it represents a missed opportunity to correctly interpret the intended use of data.

For a data translation appliance operating in dynamic contexts, as opposed to relatively static context of enterprise the most fundamental problem is *information interpretation*. What is stored in databases is not information, but *data*. However, answering questions requires information as input and output. Transition from data to information is a function of interpretation that is performed by an intelligent agent – typically human. Conversely, the problem of information interpretation is its automation, thereby removing human (as much as possible) from the process, while avoiding any interpretation errors. To that end, one must decompose the problem into fully automated and a manual *residue* tasks. The residue task must be smaller than the original task and understandable to the intended user [2].

1.3 An Alternative Approach

A desired approach would have to achieve a serviceable interoperability rapidly by automating the tasks 1.1 a-c allowing human to perform residual tasks using less effort than would be required to construct applications with other approaches. It would then allow the application to evolve toward full interoperability, but remain sufficiently open to changes. Finally, it would over time become more efficient in handling changes (Figure 2 bottom).

Semantic Web promises the economy of scale which would eventually reduce the startup effort by re-using semantic models and data transformation mechanisms. It inherits scalability and openness from the Web technology stack and adds a reach layer of metadata (data about data) which allow efficient algorithms for automating the tasks 1.1 a-c.: however, the deeper stack also contributes to disconnect between producers and consumers of data. Path-through architecture of Semantic Web protocol stack does no mitigate this problem since it only provides a syntactic shortcut from more expressive models like OWL-based to less expressive ones like RDF-based, which may result a loss in semantics. Although Semantic Web can efficiently perform such reasoning tasks as generalizing and inferring the terms not explicitly present in queries, as well as navigating and accessing resources not explicitly mentioned by users, it does not allow users to dynamically add *context* necessary for correct information interpretation [3].

An alternative approach is needed in order to skip as many intermediary steps as possible before integrated data can be accessible to user. At the same time, such quick forward approach must provide a way for user to fine-tune context for data integration early and often. Additionally, due to general purpose of DATALATOR context must represented at a sufficiently high level of abstraction to include all different types of context models used by data providers and consumers [4]. Latest research in context modeling shows that *situation* – defined as semantic abstraction from low-level contextual cues – is most appropriate context representation in wide variety of environments [5]. Two major classes of solutions – based on machine learning and formal logic – both require extensive human expert involvement in training and defining situations respectively.

In light of the above a major motivation for the approach detailed in the following sections of this report, the focus is to considerably reduce the search space for potential situations to be recognized.

2 BACKGROUND

2.1 Core Innovation

The idea at the core of this project was first formulated by the author in 2003 [6] as the following thesis:

- Producers and consumers of data must share contextual cues allowing information interpretation;
- The contextual cues are expresses by either party at the time of data production and consumption;
- Cooperation is made possible by any medium suitable for projecting the contextual cues.

While the first two statements can be easily taken for granted and do not lend themselves to any particular method of cooperation, when combined with the third statement all three can be wrapped into a specific cooperation mechanism enabled by a suitable medium. For example, if we consider printed media as a medium, then it is clear how the medium is used for projecting contextual cues required for information interpretation. In this case the printed text alone serving as content is not sufficient for the interpretation. Some additional contextual cues are required to mark beginning and end of message, and to give it a structure necessary for communication. This phenomenon is well understood in conventional media and often is stated as "The medium is the message." In some cases it is arguably impossible to separate the message from the medium.

In case of structured text data sources the medium is usually neatly separated from the content such as data contained in database. In conventional information systems medium is implemented by some type of container which provides users with contextual cues in a form of metadata. Web technologies seem to do away with very little metadata and rely on mimicking conventional media contextual cues which makes it difficult to integrate content from disparate sources. Semantic Web makes it simpler by adding a distributed form of metadata that can be accessed without a special container. However, as explained in the introduction some contextual cues can be lost in translation.

The closest analogy to context providing medium in case of Semantic Web is *ontology* – a formal logical theory that specifies the intended conceptualization of the content. Designed to be highly efficient for reasoning, ontology languages do not lend themselves easily to representing contexts. There are still no comprehensive studies on the generic and formal representation for contexts on the Semantic Web [3]. This becomes especially problematic when multiple possible contexts need to be considered, as is required for singling out one of many possible situations. In contrast, most common conventional information systems such as relational databases rely on more expressive formalisms that can be used to express constraints and dependencies characteristic of real world situations.

The need for combining open and scalable infrastructure of Semantic Web with context carrying requirements of the emerging Web-based information medium prompted the author to invent a new method of mediated cooperation – specially designed to operate alongside both conventional database systems and Semantic Web – in order to bridge the semantic gap. In 2003 Next Generation Software had filed a provisional patent application which was followed by a non-provisional application and issuing of patent US7634501 in 2009. According to the invention, mediated cooperation of consumers and producers of information occurs based on uniformed representation of real word objects as data items allocated for different semantic aspects of the objects. These data items are then connected into plausible chains of events in order to reconstruct the original context of data and to automate the information interpretation.

When applied to the challenge stated in 1.3 the new method of mediated cooperation offers a way to automatically recognize potential situations, abstracted as chains of events, directly from dependencies observed between the data items. The method provides a direct channel for low level contextual cues present in most conventional databases to percolate up to a high level user interface. Section 3 of this report will demonstrate exactly how this can be done. Moreover, as the next section will explain, the new method allows lifting data from multiple local contexts into a single unified context, where it can be efficiently integrated. Finally, Section 4 will demonstrate in a rigorous quantitative fashion that the new method considerably reduces the search space for potential situations to be recognized, making it suitable as an alternative approach for DATALATOR.

2.2 Relationship with State of the Art

Local context serves as a cooperative medium for data providers and consumers of the data source. For example, in the relational model, the data are organized into relations, which themselves are sets of data rows. Users query the data by navigating the relations in a set-oriented way. Studies have shown that queries that involve a single relation can be expressed easily by typical users, but queries that involve several relations are much more difficult and are often expressed incorrectly because the user must specify the appropriate *connections* between relations in the query. Contextual cues, such as names of relations and attributes, as well as data integrity constraints present in most relational schemas, are not sufficiently abstract for user to identify the preferred connections.

If queries need to be executed across multiple data sources (as in 1.1c) the local contexts need to be merged into a unified global context. Automating this task is one of the biggest challenges in data integration. Most approaches are focused on integrating local schemas into a mediated schema – ontology in case of Semantic Web – serving as global context for posing queries which are then reformulated into sets of queries in terms of the local contexts [7]. Increasing number of approaches focus on local context interacting with each other as autonomous nodes in peer-to-peer (P2P) networks [8]. However, all these mainstream data integration techniques are primarily concerned with correctness of query answering and data translation, as well as its resource boundedness (cost) and performance. The issues of information interpretation – discussed in 2.1 which are essential for achieving characteristics described in 1.3 – are usually not considered.

In addition, the fact that real word data sources are autonomous and often contain inconsistent data leads to several fundamental problems related to open world and closed world assumptions (OWA and CWA) which change the semantics of query answering. Basically OWA is the assumption that the truth of a statement is independent of whether or not it is stated by a data source while in its opposite for CWA, it is assumed that any statement, if not asserted true by a data source is false. The problem is that under the OWA which is used by many Semantic Web data integration systems such as Piazza and Hyperion: (i) the aggregate queries like COUNT, SUM, AVG, MAX produce counter-intuitive results [9], and (ii) some fundamental operations' complexity is NP-complete or co-NP-complete, while polynomial algorithms are available for CWA [10]. Dealing with these difficult issues shifts the focus further away from the special requirements of this project.

The more promising technique for DATALATOR is based on Universal Relation (UR) which aims to relieve the user of the need for navigating relations of a given context by assuming that *principle connections* among relations exist in that context [11]. While the price paid for that assumption is that queries may result in incorrect answers, users regain the important ability to choose connections among relations in the original local contexts. In other words, merging of local contexts required by other techniques is bypassed in UR approaches. Moreover, since no integrity constraints are present in the UR, the user has to only specify the *known* values of attributes and ask for the *unknown* ones. The data retrieved from the sources do not have to satisfy any integrity constraints at the mediated schema.

2.3 Phase I Project Requirements

As per phase I solicitation the phase 1 project is focuses on addressing the following issues:

1) Highly automated mapping. The DATALATOR must be usable by relatively untrained personnel. These personnel must be able to influence the behavior of its basic functions without knowledge of the mechanisms employed; that is, using only surface (domain relevant) information.

2) Bounded mapping effort. DATALATOR integration schemes are to be considered ephemeral. Implicit is that the longer a given source is of utility, the more effort may be spent on improving integration quality for that source.

3) Bounded answer scope. The user may only be concerned with answers to a constrained set of questions. DATALATOR will act as a "savant" providing specific answers.

4) Bounded answer quality. The user may be concerned not with optimal answers but answers that meet certain criteria for adequacy. Therefore the DATALATOR should be able to deliver "good enough" answers.

The second two requirements are essentially a counterbalance to the first two. As explained in 1.3, the goal is to achieve the 1-2 characteristics of DATALATOR, by making a reasonable trade-off with 3-4.

2.4 Phase I Project Scope

The goal of phase I was to investigate viability and design approaches for DATALATOR technology. The plan was to advance TRL3 prototype developed at Next Generation Software to TRL4 and to conduct its empirical evaluation in accordance with phase I proposal (D092-007-0197).

In phase I the development and evaluation was limited to accessing data that are stored in relational databases. Phase II will include plan for development of fully functional TRL5 prototype.

2.5 Phase I Research Platform

Stanford University Protégé open-source platform provides a plug-in architecture and Java-based API currently used by 153,642 people worldwide for semantic modeling, knowledge management and software development.

Next Generation Software had adopted Protégé as a platform for developing a research prototype named "OntoBase" for studying and testing the methods of data access and manipulation described above.

Next Generation Software had released its first publically available OntoBase TRL1 prototype in January 2008. A little over a year later the TRL2 prototype was released. By January of 2010 the TRL3 prototype was released.

3 PROTOTYPE IMPLEMENTATION

3.1 Identification of sub-problems and proposed solutions

The following illustrates key tasks that must be automated by DATALATOR, and identifies problems which are different manifestations of the information interpretation problem discussed in 1.2. In each case the residue task that requires human intervention is identified and a specific solution is described.

3.1.1 Source Modeling

Source Modeling is the process of constructing a model that best describes the intended use of data. In case of structured text data sources such as relational databases, the source is designed to hold a data from a specified domain. Specification of data is usually provided in a form of a *conceptual data model*. The most common format for conceptual data model is an entity relationship diagram. However, conceptual models like the one shown below on Figure 3 are not very useful as contexts for mediating interaction between consumer and provider of data.

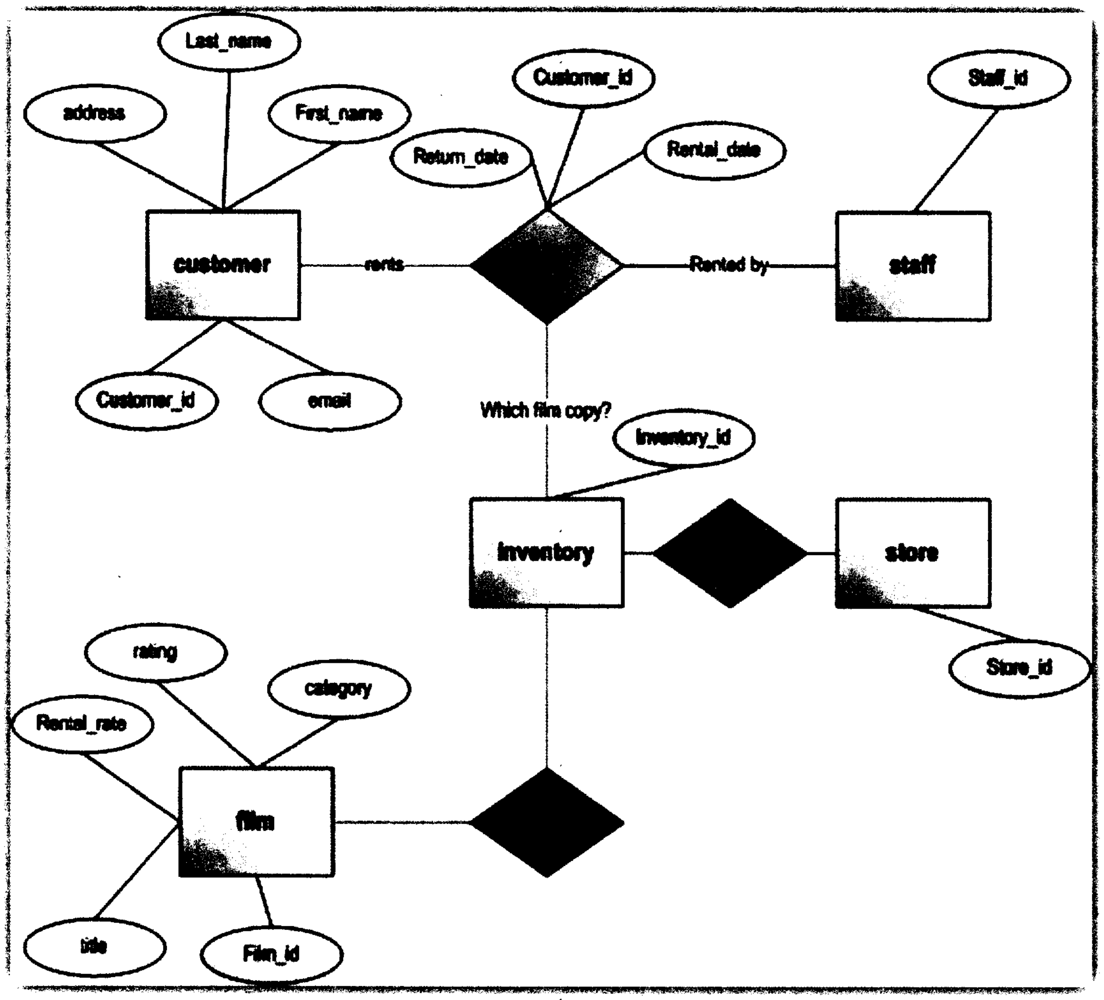


Figure 3: "*Sakila*" film database conceptual data model

As the name suggests, conceptual models help humans involved in designing data sources to better understand its intended data content, rather than intended use of that content. The intended use of data is to allow transactions and information exchange between business entities. Nothing in the conceptual model provides any contextual cues about how the transactions or information exchange can happen. The contextual cues are given to designers and users of data source separately and independently. Designers are usually given *use cases* and users are given instructions how to use application. Both depend on the application developers to connect the two sets of cues. This fact is arguably the root cause of technical barriers described earlier in 1.2. Documenting contextual cues is a function of application modeling which is supposed to be performed in a normal development cycle. The result is

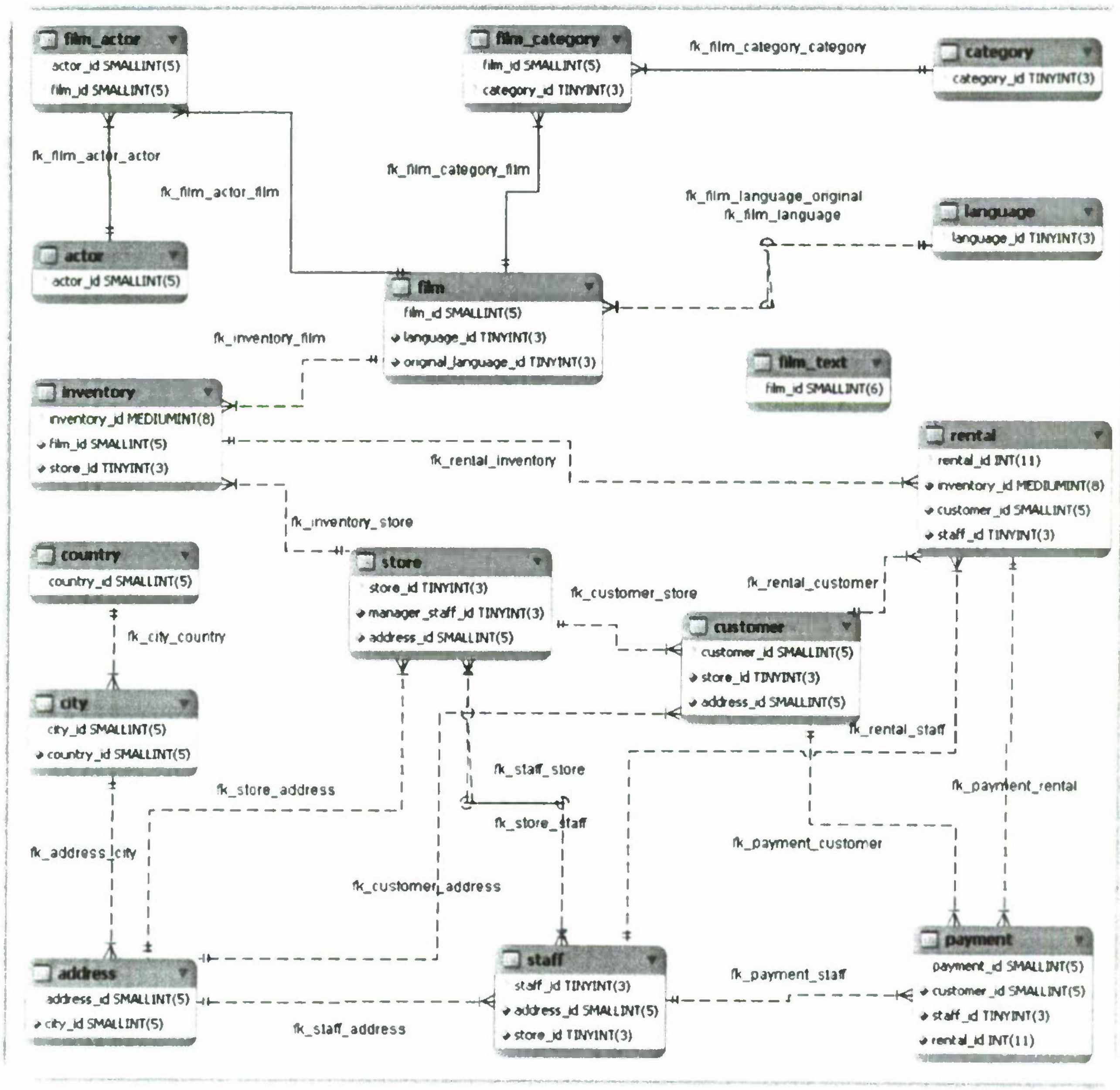


Figure 4: *Sakila* database logical data model

usually a coherent set of artifacts such as a logical data model like the one in Figure 4 and a domain model like one in Figure 5.

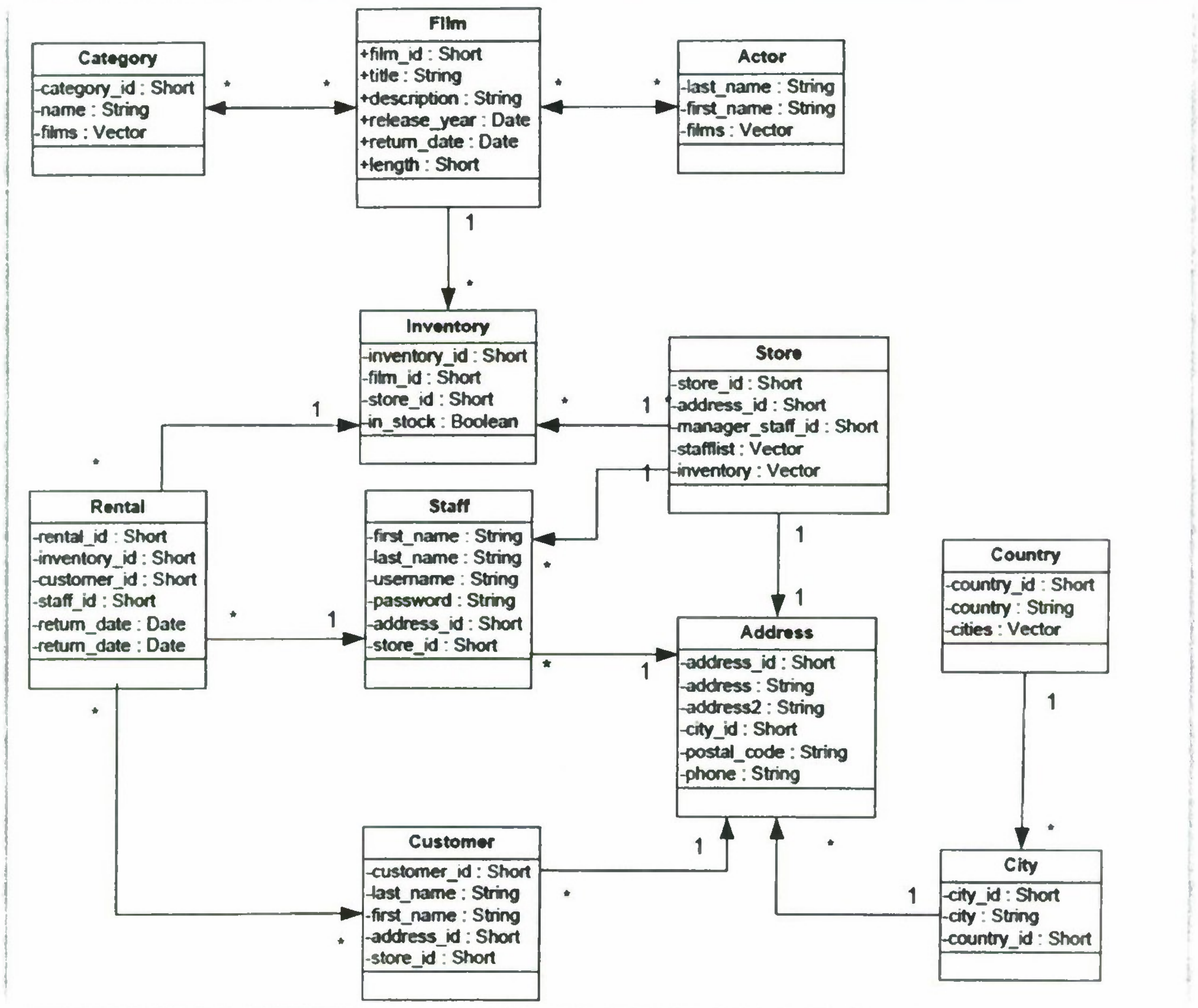


Figure 5: *Sakila* movie rental domain model

In the end when application is built and deployed along with the database, however, none of these models is provided in the application or data source context, and so it is not available to data integration utility like DATALATOR. What may be available is metadata stored in the application interface and in the database.

As discussed in 2.2, the Semantic Web significantly increases available metadata by adding ontologies. Ontology provides the means by which a rich formal, logic-based description of real world entities can be formulated. However, ontologies may be represented using different logical languages thereby potentially requiring a unifying logical framework.

The most common and best understood metadata is *schema* (Figure 6), which describe conventions for structuring, typing, and naming data. However, most schema definition languages such as database schema or XML lack the level of abstraction and expressive power required to convey the meaning and consequences of data it describes. Limited semantics of schema combined with incompatible schema dialects requires a *mediator* to provide a data over different schema dialects [12]. Moreover, schema needs to be augmented to accommodate additional semantics required for correct information interpretation.

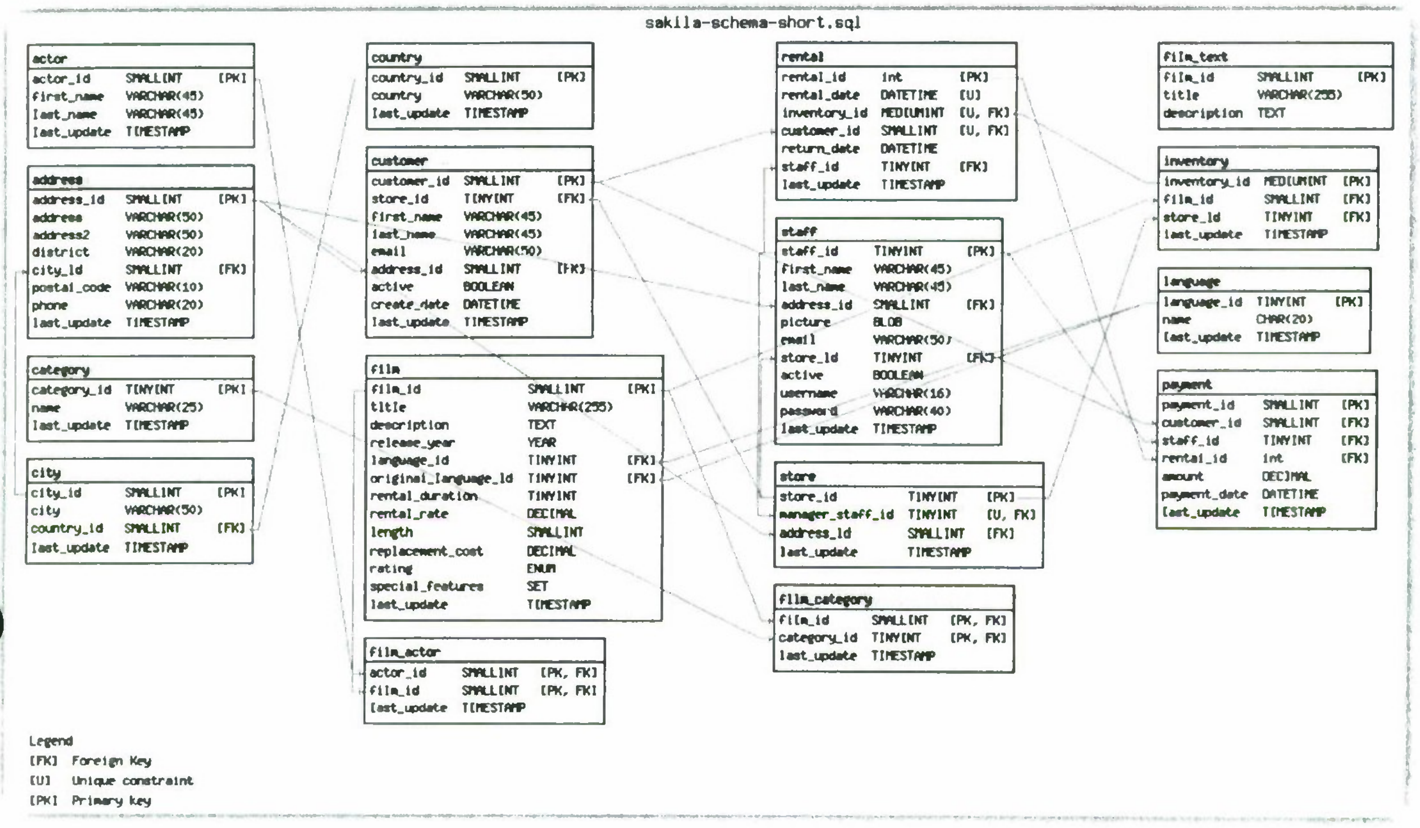


Figure 6: *Sakila* database schema

3.1.1.1 Problem

Full context required for information interpretation is not provided by most data sources and needs to be recovered from available metadata. This can be a time consuming effort that requires knowledge of technical artifacts like database schema (Figure 6). Constructing a model that can be used for automated information interpretation may become a "knowledge acquisition bottleneck" in the process of data source integration. Low level contextual cues buried in metadata cannot be replaced with ontology that uses less expressive language than the language used to express metadata. Therefore, automatically generating ontology from metadata such as relational schema can result a loss in semantics. The recovery of lost semantic information, as well as further enrichment of semantic model needs to be highly automated but at the same time allow direct users' involvement in order to dynamically add task-specific context not present in data source metadata.

3.1.1.2 Proposed solution

The author proposed a solution [13] that allows automated enrichment of data source schemas extracted in a form of *Star Diagrams* (Figure 7). The method generates all legal Event-Condition-Action (ECA) chains (Figure 9), and organizes it into highly reusable micro-theory called *causal stream*.

- Automated task: Translate metadata from different schema dialects into Star Diagrams, and then automatically generate all the legal ECA chains of events.
- Residue task: Allow user to augment and modify Star Diagrams.

3.1.1.3 Technical details

A Star Diagram is a simple graph that reflects all functional dependencies of a single data item. For example, in Figure 7 arrows indicate that a rental cannot take place unless *inventory, staff* and *customer* are all known. Star Diagrams can be used as a "common denominator" for extracting data models from heterogeneous data sources.

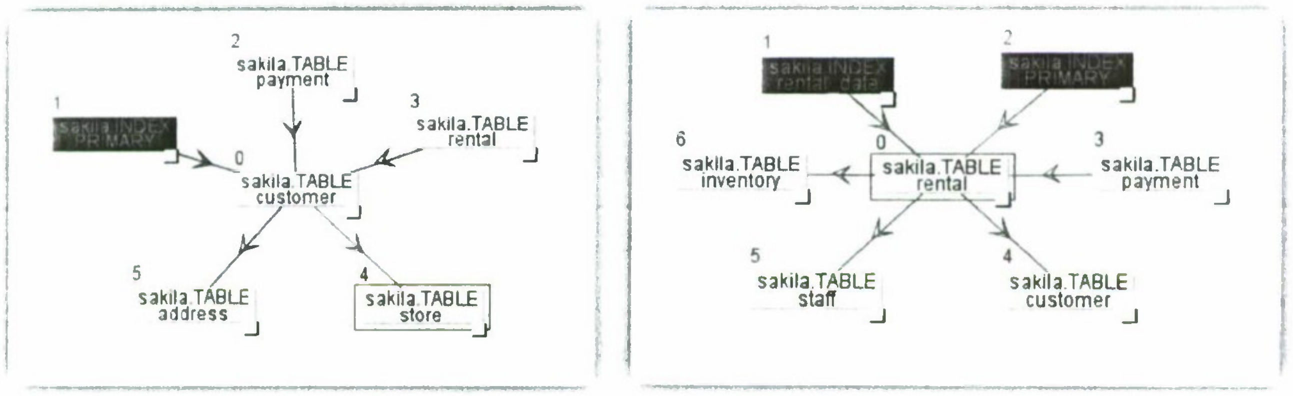


Figure 7: Star Diagrams generated from *Sakila* schema

When applied to the source modeling problem, the proposed solution automates dicovery of semantic model that best represents the intended use of data contained in the source. This is accomplished by presenting a user with a hypothetical process represented by the flow of data (Figure 9).

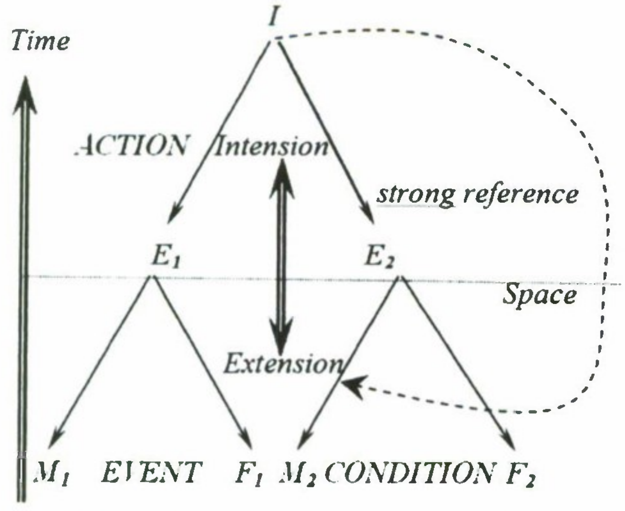


Figure 8: Causal basis for strong reference

Each hypothesis is based on a possible sequence of ECA steps or chains of events. It is based on causal framework that includes four types of cause or explanation [14]:

E efficient - that makes a change happen,
M material - what the change happens to,
F formal - what the change results in, and
I final - the end or purpose of the change.

The basic assumption that no event can cause itself reduces the number of possible arrangements of chains to managable size (subject to experimental evaluation, see 4.4.2).

Figure 9 show a possible ECA chain generated by combining Star Diagrams extracted from the Sakila database schema. Algorithm used for that purpose by OntoBase is treating all functional dependencies as directed binary "part-of" relations connecting database records into part-whole hierarchy. Which one of four types of causes is assigned to each record determines the intended used of data in a particular situation. The records that reflect *events conditions* and *actions* of a given situation are marked as M or F. The records that do not directly reflect the situation are marked as E. According to the proposed method only the records marked as M or F can be promoted from local context to global context via an "is-a" relation connecting intensional and extensional sides by *strong reference* [14], as described in more detail later.

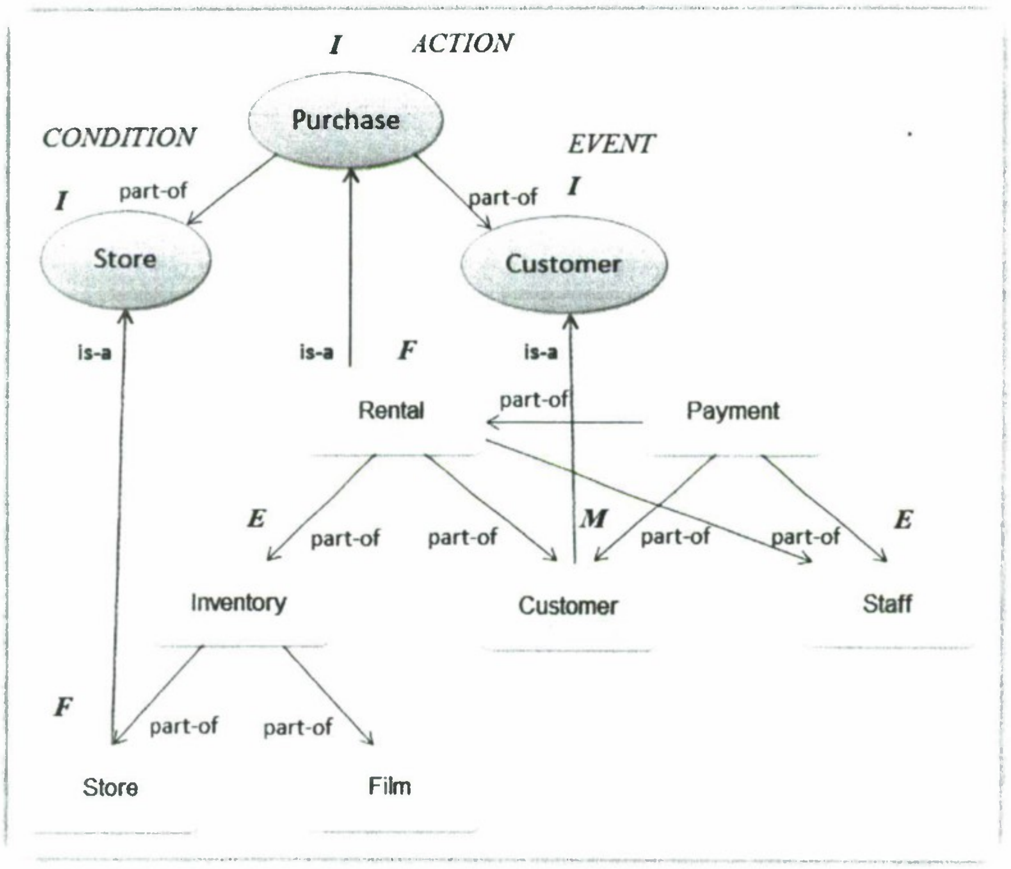


Figure 9: Event-Condition-Action (ECA) chain for film rental event

3.1.2 Data Integration

Data Integration is translation of data from multiple sources and its subsequent integration to answer queries in a global context. Unlike the source modeling that deals with sources' metadata, data integration deals with actual data such as rows in database or more complex data objects which represent real world entities. The goal is to integrate data across multiple data sources, so that references to the same entity can be resolved. When integrating an ad-hoc data source one can no longer rely on a *unique name assumption* such that different names always refer to different entities. This necessitates identifying any co-referent names and subsequently rewriting them to a common form.

3.1.2.1 Problem

In the absence of a mechanism that would guarantee universal referential integrity, heuristics and machine learning approaches were proposed in vein of "best-effort" [15]. These techniques make extensive use of context information (such as associations between references) to provide evidence for reconciliation decisions [16]. However, the resulting referential integrity heavily depends on the context implicit in references, as opposed to the context in which the data is later being used.

3.1.2.2 Proposed solution

The author had proposed [13] a method that allows user to define a global context of the performed task by simply navigating a *situation tree* (Figure 11) that was automatically generated in response to user selecting a target element from a local context (Figure 10).

- Automated task: Generate a situation tree by applying causal constraints to ECA chains in 3.1.1.3.
- Residue task: Navigate the situation tree in order to interactively define context of a particular task.

3.1.2.3 Technical details

The proposed solution has been implemented by Next Generation Software in a form of OntoBase tab plugin-in to Stanford University Protégé platform .

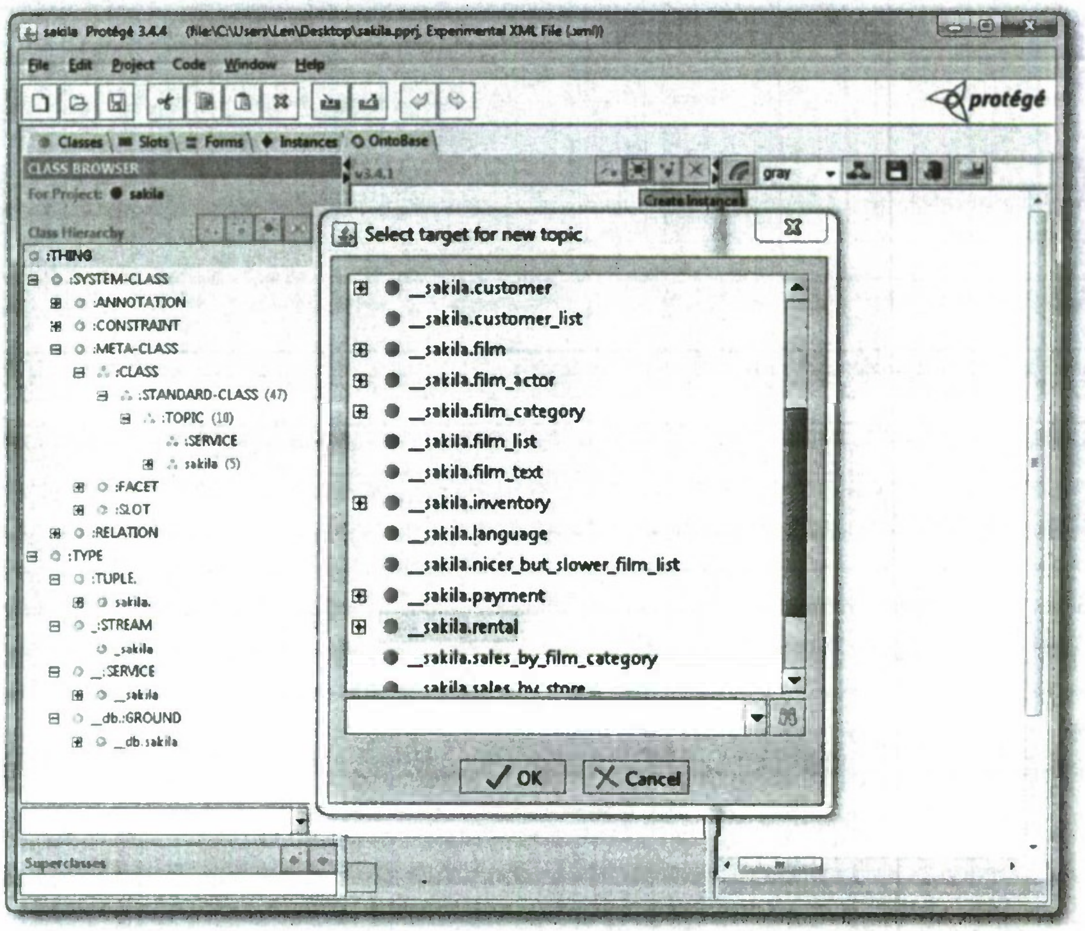


Figure 10: OntoBase plugin running inside Protégé editor

Upon launching the plug-in allows users to connect to any relational database and import its metadata into Protégé in a form of ontology generated specially to represent local context of the database. A user can then begin a guided generation of a global context divided into mediating containers as discussed in 2.1. Each container is a frame that corresponds to Protégé abstract class and can have concrete subclasses representing real word situations. Causal streams made of ECA chains in 3.1.1.3 are generated on-demand when user selects a target from local context (Figure 10). In response all possible ECA chains leading to a requested target will be generated. Because of causal constraints the number of chains will be manageable, and response time will be acceptable (see 4.4.2). All chains are presented to a user in a form of a situation tree (Figure 11). This simple and intuitive user interface allows the user to quickly identify the context in which data is being accessed for a particular task. Once the situation tree path is selected it defines a situation stored in OntoBase-generated subclass of the container.

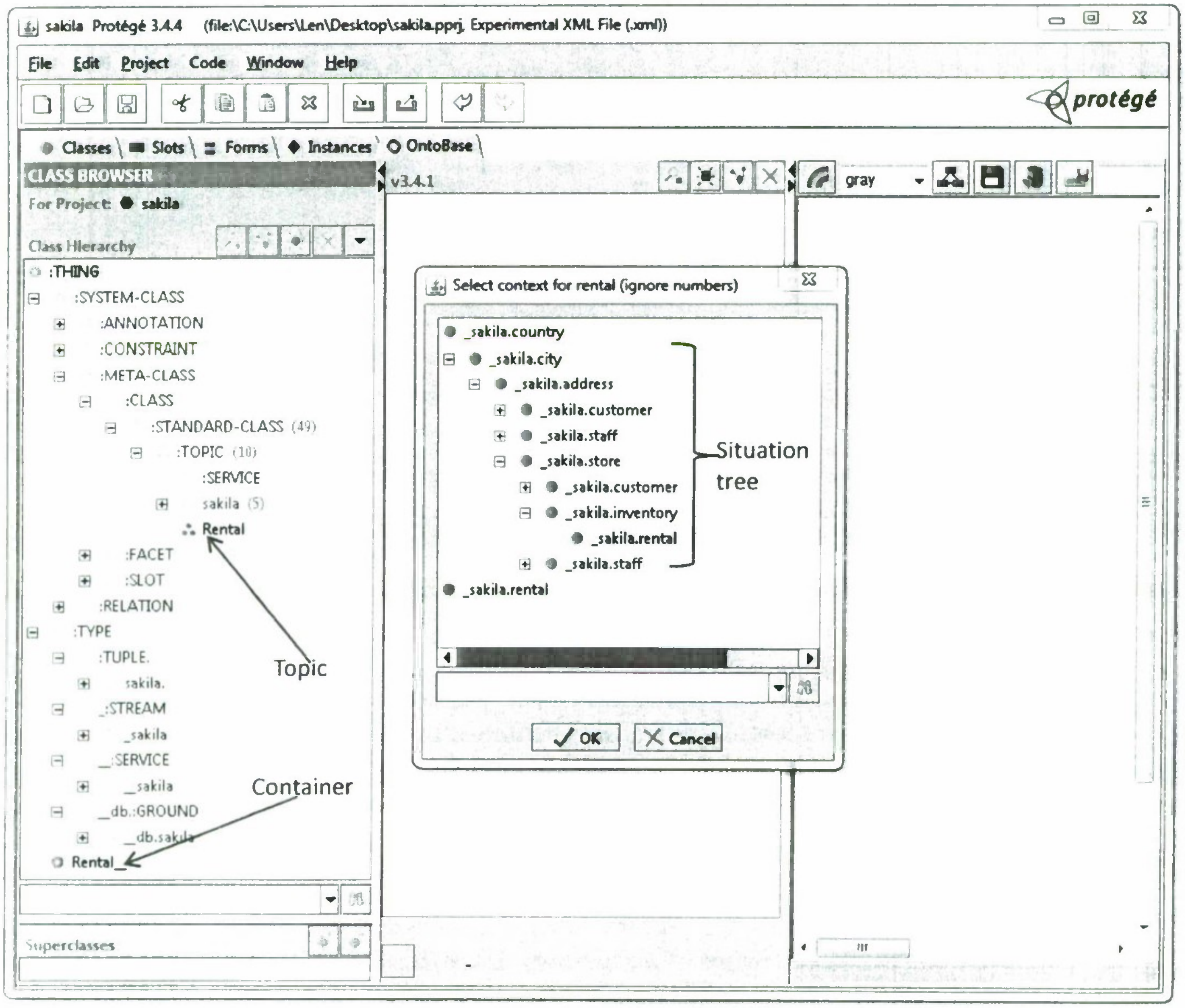


Figure 11: OntoBase situation tree for *Rental*

Figure 11(left) also shows the beginning of global context generated by OntoBase in response to user selections. The global context consists of abstract topics and containers that can be used to pose user queries. Figure 12 shows a UR comprised of all attributes found on the preferred path selected by user in the situation tree. The user can select any subset of attributes in UR to represent a particular situation. The selected attributes represent the *unknown* part of the situation as explained in 2.2. The *known* part is defined in the next step when user is presented with automatically generated situation

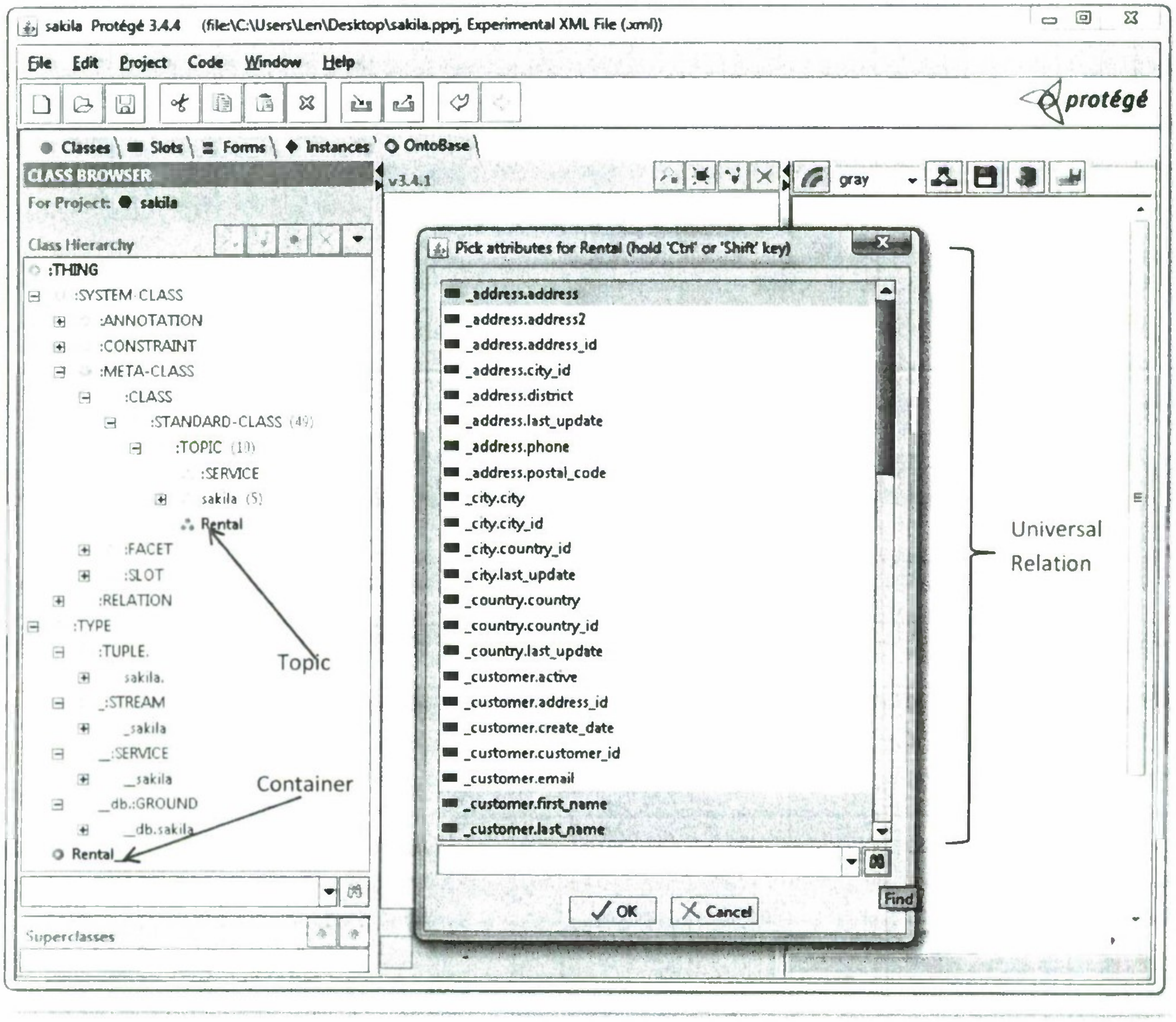
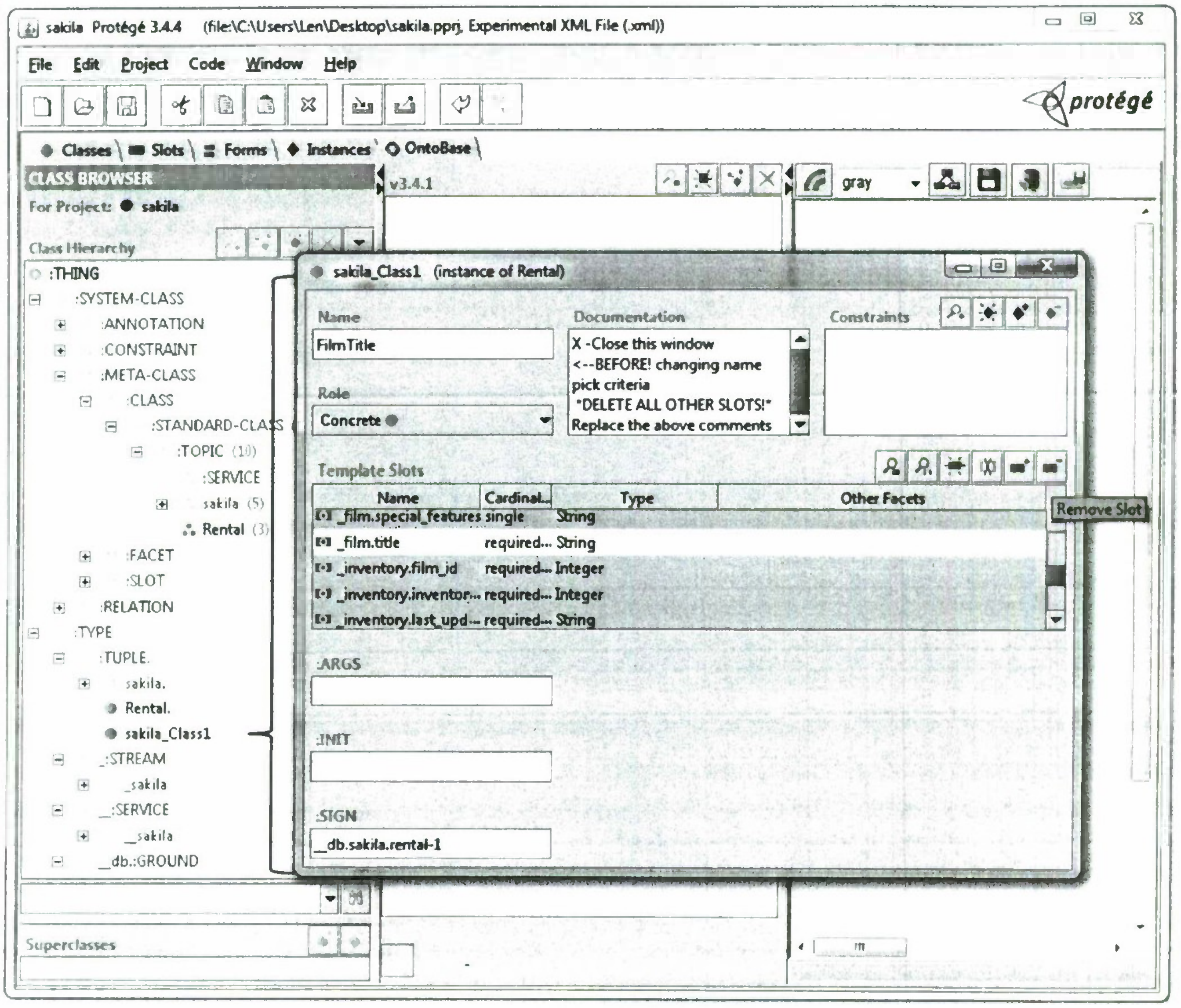


Figure 12: Universal Relation for *Rental* topic

class (Figure 13). In this example of film rental situation a film title will be used to find a film for rental. One of the interesting features of Protégé frame editor is that a class can be an instance of another class that is called meta-class. Topics are meta-classes that have situations as instances. Situation classes in turn can have instances that represent ECA chains (Figure 9). OntoBase uses topic meta-classes as a mechanism for promoting data references from local context to global context, as explained in 3.1.1.3. This allows OntoBase to represent the same situation locally and globally as Protégé classes and meta-classes respectively. Essentially OntoBase allows any data source to be treated as a situations' store.



sakila Protégé 3.4.4 (file:\C:\Users\Len\Desktop\sakila.pprj, Experimental XML File (.xml))

sakila_Class1 (instance of Rental)

Name: FilmTitle

Documentation:
X -Close this window
<--BEFORE! changing name
pick criteria
"DELETE ALL OTHER SLOTS!"
Replace the above comments

Constraints

Role: Concrete

Template Slots

Name	Cardinal...	Type	Other Facets
_film.special_features	single	String	
_film.title	required...	String	
_inventory.film_id	required...	Integer	
_inventory.inventor...	required...	Integer	
_inventory.last_upd...	required...	String	

:ARGS

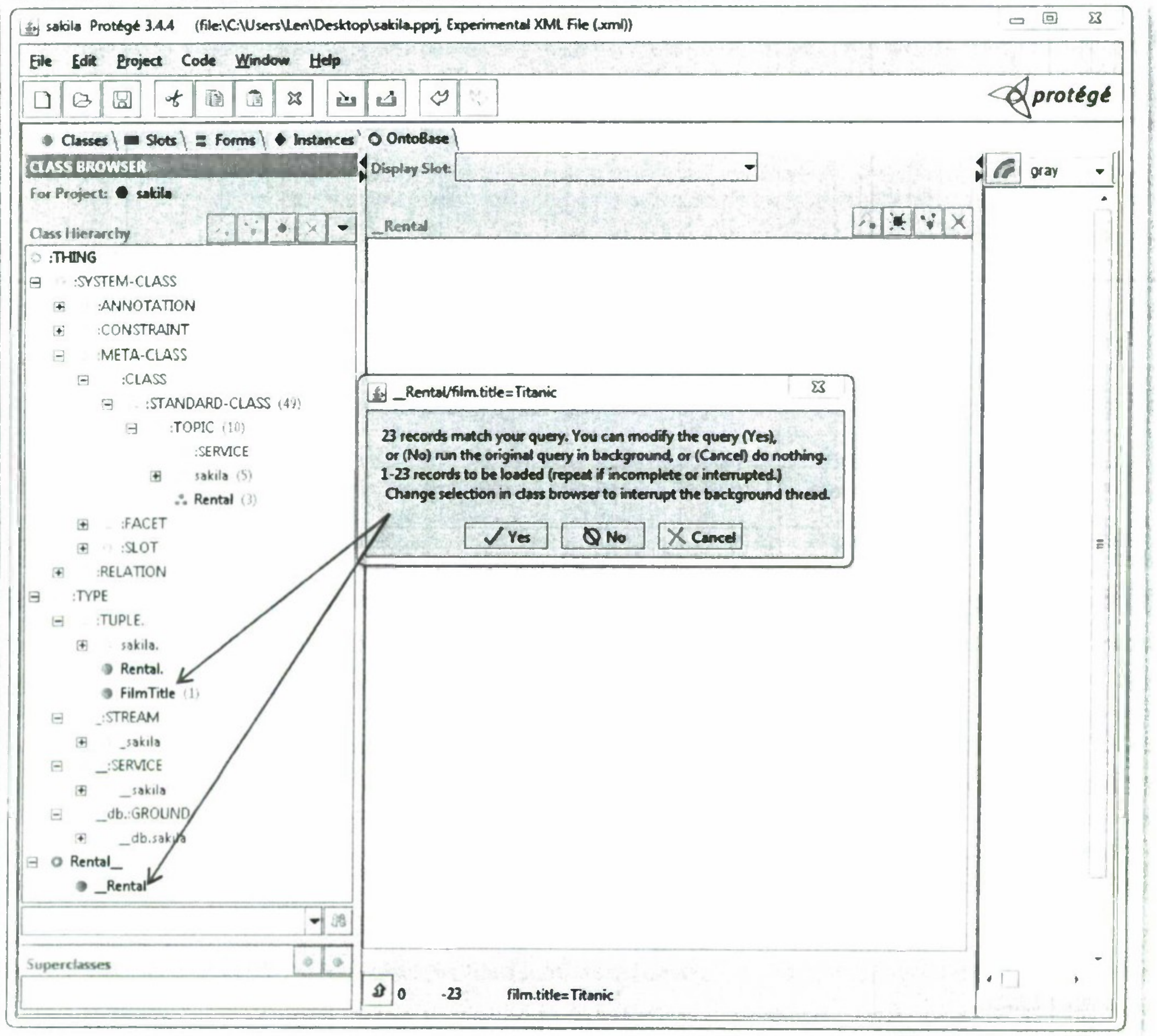
:INIT

:SIGN: __db.sakila.rental-1

Figure 13: Generation of a situation instance class of *Rental* topic meta-class

Figure 14 shows result of query atomaticaly generated by OntoBase in responce to a user providing a known part of Rental situation as: __Rental/film.title=TITANIC. Intenraly OntoBase generates and executes SQL query:

SELECT count(*) FROM sakila.rental join sakila.inventory on rental.inventory_id=inventory.inventory_id join sakila.film on inventory.film_id=film.film_id join sakila.language on film.language_id=language.language_id join sakila.customer on rental.customer_id=customer.customer_id join sakila.store on inventory.store_id=store.store_id join sakila.staff on store.manager_staff_id=staff.staff_id join sakila.store as fk_staff_store on staff.store_id=fk_staff_store.store_id join sakila.address on store.address_id=address.address_id join sakila.city on address.city_id=city.city_id join sakila.country on city.country_id=country.country_id join sakila.address as fk_staff_address on staff.address_id=fk_staff_address.address_id join sakila.address as fk_customer_address on customer.address_id=fk_customer_address.address_id join sakila.staff as fk_rental_staff on rental.staff_id=fk_rental_staff.staff_id join sakila.store as fk_customer_store on customer.store_id=fk_customer_store.store_id WHERE film.title='TITANIC'


sakila Protégé 3.4.4 (file:\C:\Users\Len\Desktop\sakila.pprj, Experimental XML File (.xml))

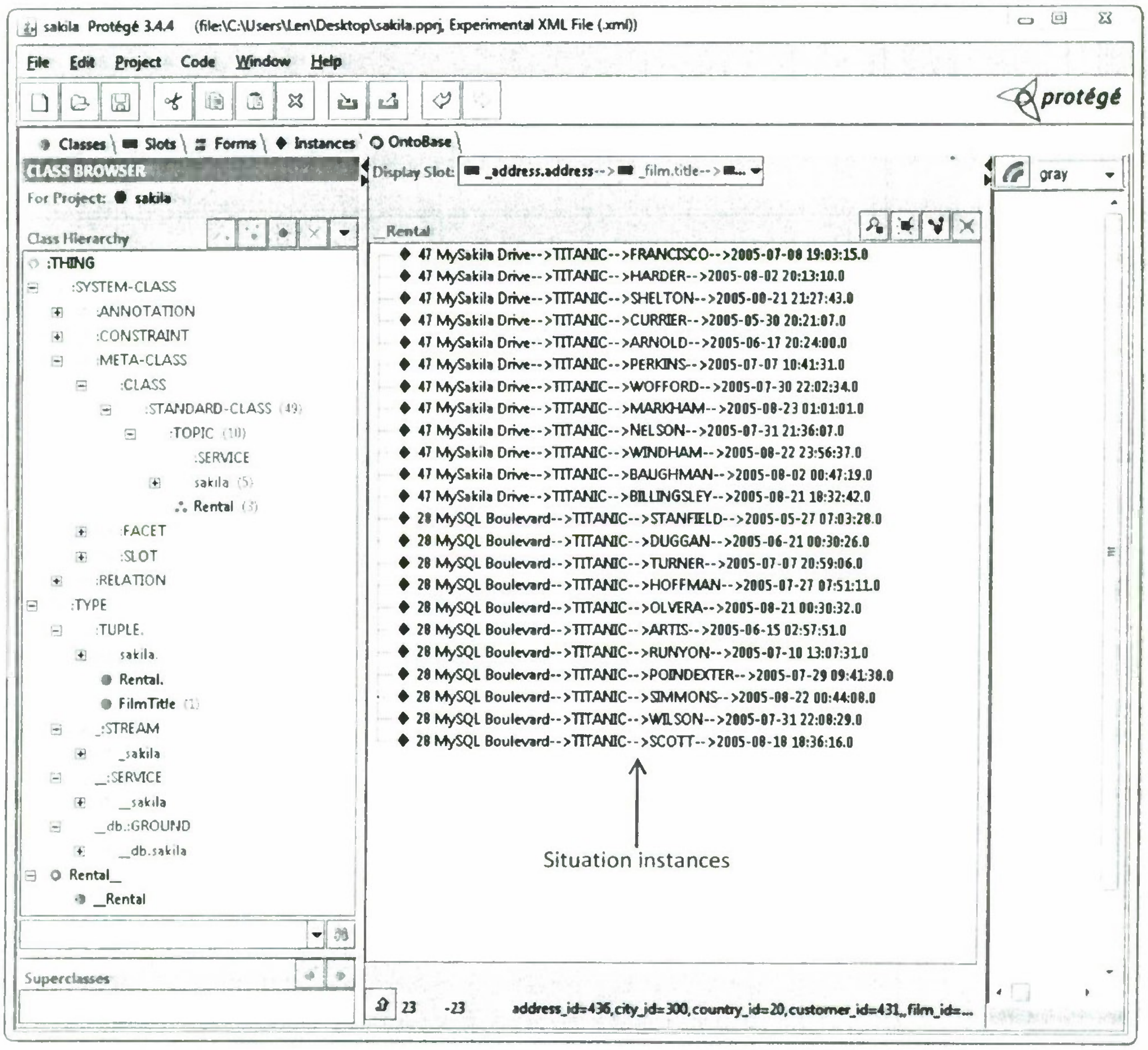
_Rental/film.title=Titanic

23 records match your query. You can modify the query (Yes), or (No) run the original query in background, or (Cancel) do nothing.
1-23 records to be loaded (repeat if incomplete or interrupted.)
Change selection in class browser to interrupt the background thread.

0 -23 film.title=Titanic

Figure 14: Matching *Rental* situations in database

The SQL query above performs joins on all functional dependencies (eg. nodes 4,5,6 connected with red lines on Star Diagram in Figure 7). Note that not every functional dependency is a *priciple* connection in UR (see 2.2). OntoBase uses aliases to mark joins that are not on the path selected by user in the situation tree. Figure 15 shows the resulting situation instances found in the database. Each instance has unique name composed of key values from functional dependencies.



```
47 MySakila Drive-->TITANIC-->FRANCISCO-->2005-07-08 19:03:15.0
47 MySakila Drive-->TITANIC-->HARDER-->2005-08-02 20:13:10.0
47 MySakila Drive-->TITANIC-->SHELTON-->2005-08-21 21:27:43.0
47 MySakila Drive-->TITANIC-->CURRIER-->2005-05-30 20:21:07.0
47 MySakila Drive-->TITANIC-->ARNOLD-->2005-06-17 20:24:00.0
47 MySakila Drive-->TITANIC-->PERKINS-->2005-07-07 10:41:31.0
47 MySakila Drive-->TITANIC-->WOFFORD-->2005-07-30 22:02:34.0
47 MySakila Drive-->TITANIC-->MARKHAM-->2005-08-23 01:01:01.0
47 MySakila Drive-->TITANIC-->NELSON-->2005-07-31 21:36:07.0
47 MySakila Drive-->TITANIC-->WINDHAM-->2005-08-22 23:56:37.0
47 MySakila Drive-->TITANIC-->BAUGHMAN-->2005-08-02 00:47:19.0
47 MySakila Drive-->TITANIC-->BILLINGSLEY-->2005-08-21 18:32:42.0
28 MySQL Boulevard-->TITANIC-->STANFIELD-->2005-05-27 07:03:28.0
28 MySQL Boulevard-->TITANIC-->DUGGAN-->2005-06-21 00:30:26.0
28 MySQL Boulevard-->TITANIC-->TURNER-->2005-07-07 20:59:06.0
28 MySQL Boulevard-->TITANIC-->HOFFMAN-->2005-07-27 07:51:11.0
28 MySQL Boulevard-->TITANIC-->OLVERA-->2005-08-21 00:30:32.0
28 MySQL Boulevard-->TITANIC-->ARTIS-->2005-06-15 02:57:51.0
28 MySQL Boulevard-->TITANIC-->RUNYON-->2005-07-10 13:07:31.0
28 MySQL Boulevard-->TITANIC-->POINDEXTER-->2005-07-29 09:41:38.0
28 MySQL Boulevard-->TITANIC-->SIMMONS-->2005-08-22 00:44:08.0
28 MySQL Boulevard-->TITANIC-->WILSON-->2005-07-31 22:08:29.0
28 MySQL Boulevard-->TITANIC-->SCOTT-->2005-08-18 18:36:16.0

23   -23   address_id=436,city_id=300,country_id=20,customer_id=431,,film_id=...
```

Figure 15: Instances of *Rental* situation

In order to recognize situations in the records fetched by OntoBase an additional step is required: the user must pick some attributes that would serve as global descriptors to distinguish one situation form another. Protege editor has a convenient feature called "Display Slot" (Figure 15 top) that can be used for this purpose. Users of Protege can pick any sequence of instance slots to be displayed instead of instance name. In addition users can place any text between the slot values to mark transitions between events in that situation. It is up to user to decide what set of attributes makes the situation distinct. This mechanism combined with selecting of the preferred path in the situation tree allows user to establish the intended use of data. For example, if user has picked slots _address.address, _film.title, _customer.last_name and _rental.rental_date, then each instance of _Rental is recognized only by the sequence of these fields. Note that address is an indirect functional dependency of rental, but there are three possible functional paths between address and rental passig through customer, staff and store. Which address is used by UR depends on what path was selected by the user in the situation tree. According to user selection (Figure 11) the path connects rental node to address node via store, which makes it a principle connection in this particular situation. The intented use of data in this case is to represent rental events that took place at a store address.

Data integration can now be done by recognizing the same situations acros multiple data sources and eliminating or resolving unwanted duplcates. The described procedure is logicaly equivalent to creating a global schema or ontology, mapping data sources to the global schema and then performing query and data transaltion. However, the main difference is that no expert involvement was necessary as the end user could directly interact with local context and construct a global context that represents the intended use of data. This allows DATALATOR to skip as many intermediary steps as possible before integrated data can be accessible to user, as suggested in 1.3. In addition the proposed method provides a logic for reasoning about identity of objects represented in data. In most logic frameworks identity is intensional, that is given to objects by definition or *intension*. But in causal stream, the identity of each event can be calculated based on the identities of all preceding events, that is by *extension*. As a result, the events can be referenced by extension. This means that when reasoning about objects in a context of a particular situation, the identity of each object has two components, namely: 1) intension and 2) extenstion, connected by the context. This connection is called *strong reference* [6].

Strong reference provides a mechanism by which the referential integrity can be achieved in context of a given situation, as opposed to universal referential integrity that is difficult to achieve. Strong reference is different from a global reference such as Universal Resource Identifier used in Semantic Web in that it actually gives semantics to reference. This helps to avoid counterintuitive results that often arise in data integration. In fact giving proper semantics to questions and answers in global contexts is one of the main problems in data integration. However, establishing strong references requires maintaining global contexts for all relevant situations. Each global context established under the strong reference semantics provides a mediating container as discussed in 2.1. And since the strong references enabled by such container are endowed with natural semantics, it can be called *Semantic Type Container (TC)*. OntoBase effectively tums each situation container into a TC. If used on a global scale it can facilitates the emergence of semantically reach medium in which multiple contexts can interact and evolve. Situation ontologies semi-automatically created with OntoBase are in essence micro-theories of context and can be used for inferring new knowledge from results of performing data integration tasks. This can be a solution to knowledge acquisition bottleneck.

3.1.3 Alignment and Mapping of Models

As demonstrated in 3.1.2 data integration can be achieved by resolving references to the same situations (intended use of data) across multiple data sources. As discussed in 3.1.1 situations are recognized by human (or other intelligent agent) in the original local context of data source, relying on available meta-data. Discovering the same situation in different local context is in fact a basis of contracting and maintaining a global context. However, autonomous data sources may use different vocabularies and can store data from different domains. In that case, extracting Star Diagrams from meta-data (Figure 7) may result in incompatible situation trees (Figure 11). This is a "chicken-and-egg" problem that requires *a priori* context for reconciling references, while *a posteriori* context is established by making new references. The standard approach to this problem is for domain experts to write *bridging axioms* [17] to reconcile the differences between data sources. This high "price of admission" does not decrease over time, as changes and new data sources are introduced.

An alternative to the standard approach is treating data sources as autonomous agents and providing a cooperative medium in which the agents can agree on common interpretations within contexts of given tasks. This general type of interoperability is known as *Emergent Semantics (ES)* [18]. One possibility of how this might be achieved is currently being opened in the area of peer-to-peer data management systems (PDMS), where the peers entering into agreements are called *acquaintances* [19]. However, use of direct schema mapping between the data sources as an attachment law is equivalent to the aforementioned bridging axioms. In order for peers to act as cooperating agents, some logical inference must take place inside each peer. Strong reference that gives semantics to all references (internal and external to agent) makes this possible

Using common topic meta-classes described in 3.1.2.3 as shared vocabulary spaces could be seen as an attachment law of ES networks which start from a small nucleus of nodes, and expand with the arrival of new nodes. Since topics are meta-classes, used as stereotypes when generating semantic models on-demand, no direct mapping between the models is required. Employing topics as lexical resources enables semantic agility and allows various agents cooperate at a higher level of abstraction.

3.1.3.1 Problem

Achieving semantic agility is a challenge of maintaining agile semantic mappings across different data sources, so that slight changes such as adding or removing individual data sources do no result in a significant downtime. In other words – having too rigid semantic mappings would make DATALATOR brittle in dynamic operating environment, hence another name that is often used for this challenge – "brittleness problem" [20].

3.1.3.2 Proposed solution

OntoBase is designed to give cooperating agents additional autonomic reasoning capabilities reinforcing conventional frame-based representation by direct grounding of frames to data sources. The grounding is achieved by using frame *facets* as data binding mechanism.

- Automated task: Generate *ground* frames (ontology) with facet-based data binding to data source
- Residue task: Assign semantic types (stereotypes) to situations derived from the ground frames.

3.1.3.3 Technical Details

If all metadata is put aside, each data source can be treated as a collection of data items called *tuples*, an ordered sequence of typed values given a name. Named tuples are also a basis for so called associative memory architecture known as *tuple space* [21]. The notion of tuple space is a complete opposite of ontology in a sense that it does not establish any relations between the named tuples. So, one can attach any meaning to tuple names without violating any existing semantic model, such as database schema. This observation is a basis for the proposed solution: to replace the rigid connection between data and schema (inside a data source) with a much more flexible one established outside the data source by a TC.

Context-specific stereotypes that the user assigns to tuples need not go into any source or target ontology. Rather, these stereotypes act as meta-modeling primitives. When implemented in Protégé stereotypes are defined as meta-classes for all class frames in ontologies. Since the stereotypes are assigned by user directly to tuples, regardless of the original data source schema, they form a *Semantic Overlay Network (SON)*, in which the regular classes are instances.

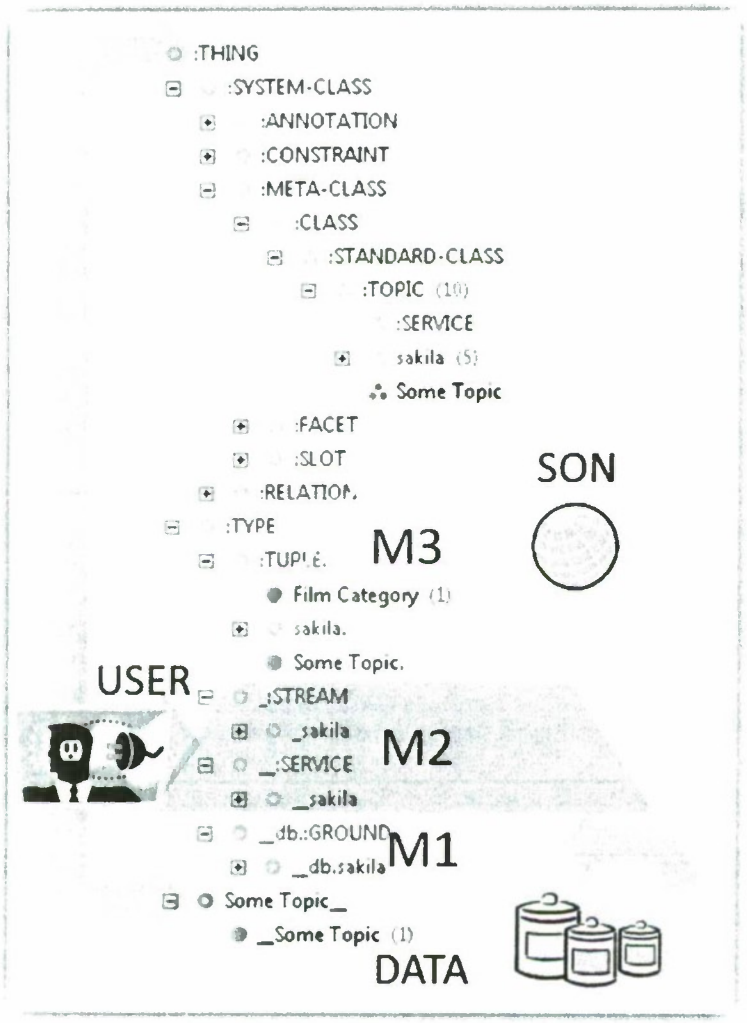
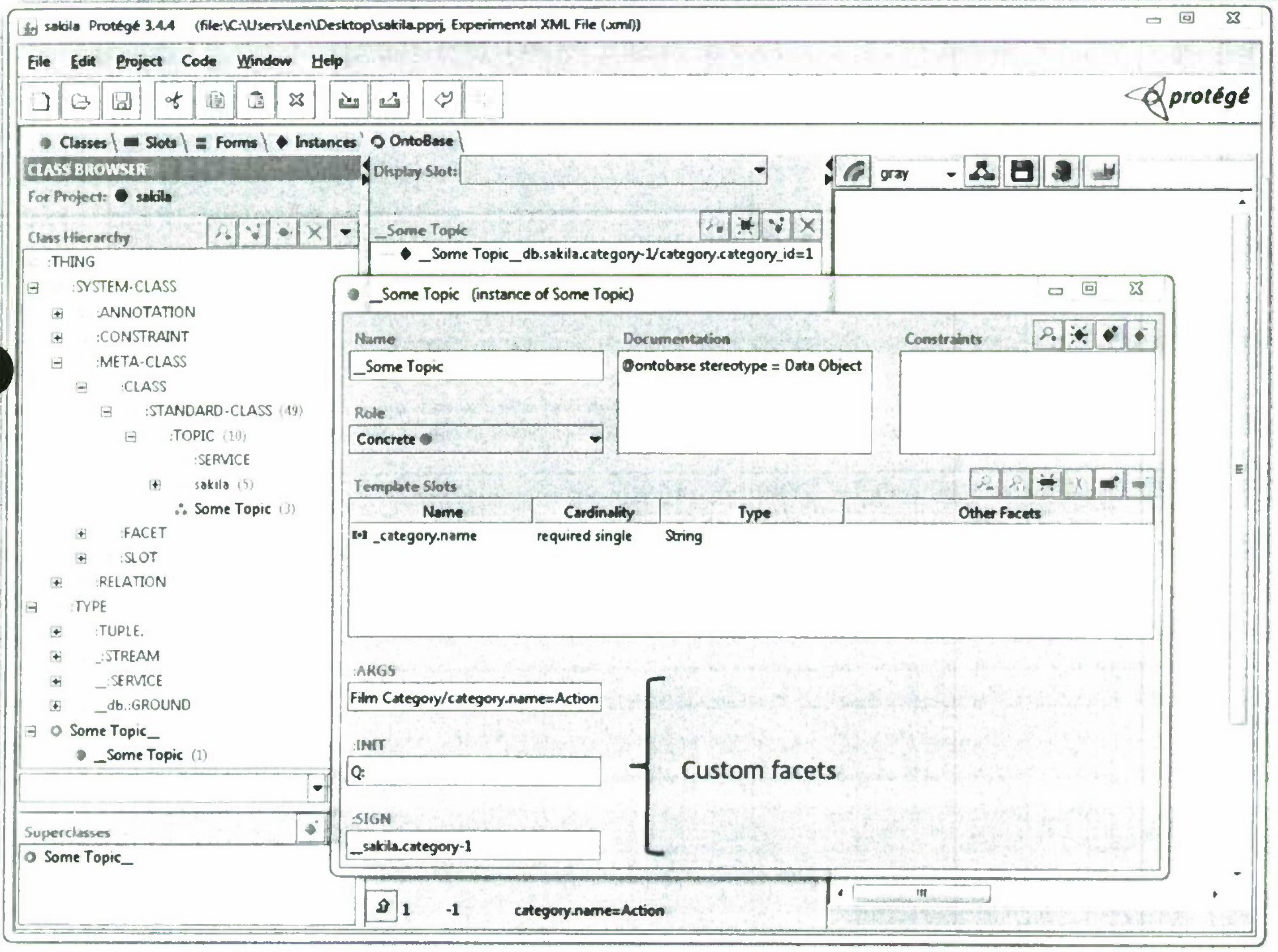


Figure 16: OntoBase logical organization

According to the proposed method: schema is extracted from data source and is used to automatically generate a ground ontology, which later serves as a proxy to the schema (Figure 16) Data is automatically bound to the ground ontology in such a way that the tuples of data can be retrieved and given unique identifiers linking it to the data in the original source. In essence the schema is virtually "lifted" from data source and placed into the ground ontology at M1 level. At the same time, additional class frames are generated at M2 level that provides an abstract representation of the source as a collection of data services. At the highest level of abstraction M3 more class frames are generated that model both data and metadata as named tuples. Names of frames at all three levels are derived from the names of schema elements by adding string prefixes and postfixes.

When user defines new topics as described in 3.1.2.3, new class frames are automatically generated at all three levels, with names similarly derived from the topic name (e.g. *Some Topic*). M1 level is bound to the data source schema, while M2 and M3 are mapped to M1 by means of custom facets (Figure 17). M2 frames serve as a mapping layer from the generated situation classes (e.g. _*Some Topic*) to M1 frames that translate queries and transform data fetched from data sources. Each situation class has three OntoBase facets which store the mappings (Figure 17). In addition each template slot of the situation class has four more OntoBase facets storing attribute metadata such as data transformation rules. The generated situation classes can be manually extended to include user-defined slots with OntoBase facets storing expressions for computing the attribute values from other data-bound attributes. Moreover, using frames allows situation classes to encapsulate behavior rules and constraints. For example, in Protégé frames can be used as data edit form with custom widgets and validation. This allows situation classes to act as business objects and to be linked to other situation serving as domain objects. However, situation classes remain bound to M1 layer which is in turn bound to data sources.

More generally, M1 plays role of Platform-Specific Model (PSM). M2 plays role of Platform-Independent Model (PIM) which encompasses local context of data sources, but is not directly dependent on the platform such as database technology or data formats. In other words, there is a clear air gap between the two layers which allows M2 to serve as PIM. The only downward connection to M1 exists entirely outside OntoBase in a form of TC (e.g. *Some Topic_*) which are automatically generated super-classes for the situations as described in 3.1.2.TCs do not carry any OntoBase facets and thus are completely independent from it. In fact all of them are abstract and generated as direct sub-classes of the Protégé root class. This allows containers to be freely detached from ground ontology. At the same time in virtue of having concrete situations as their subclasses the TCs can carry the data loaded into the situation classes and allow users to access the data in a platform-independent fashion.



Name: _Some Topic
Documentation: @ontobase stereotype = Data Object
Role: Concrete

Name	Cardinality	Type	Other Facets
_category.name	required single	String	

:ARGS: Film Category/category.name=Action
:INIT: Q:
:SIGN: _sakila.category-1

1 -1 category.name=Action

Figure 17: Situation class frame in OntoBase

Finally, the M3 layer plays the role of meta-model for both M1 and M2 by providing modeling primitives that are not bound to local contexts. Frames in M3 represent data tuples and, as explained at the start of this section, are not constrained by any lower level semantic model. Nevertheless, these tuples can carry the same data as do instances of M1 and M2 classes. For every new topic a single M3 class frame is generated (e.g. *Some Topic.,* note the "." postfix which the topic meta-class name does not have). M3 instances carry only the key attributes of the respective M1 and M2 instances. Because M3 frames are not dependent on each other, they can be exchanged between different context and play the role of the global data space or tuple space. However since tuple names and attributes are shared with M2 classes at the lower levels of abstraction, identity of each tuple is tied to identity of objects it represents. Moreover, as already mentioned in 3.1.3.2 tuple identity has intensional and extensional components. Its *intension* is a set of attributes which are globally unique, and its *extension* is a set of all lower level objects associated with it. If the objects retrieved from different local contexts are associated with the same tuple, then these objects are treated as mutually interchangeable in corresponding global context. Therefore, the lower level objects are connected to tuples by *is-a* relation (vertically) and by part-of relation to each other (horizontally) within the same local context (Figure 9). The tuples, on the other hand can flow freely across the multiple global contexts and carry with them the intensional components of the object identities. This provides for a very natural and intuitive attachment law in ES architecture [13]: no object can be a part of itself. No *acquaintances* can be established in violation of this basic logic constraint; and because tuples are now globally constrained, they form a semantic type system.

The described above logical organization of OntoBase makes it a suitable cooperation medium as introduced in 2.1. The strong reference is enabled globally (i.e. across multiple contexts) by the tuple type system of each global context. At the same time the strong reference can be made to objects residing in local contexts, with one significant limitation. Each strong reference can refer to more than one object. More formally the strong reference is defined up to isomorphism and the objects it identifies form a bicategory. Less formally the TC architecture based on strong reference provides a way to logically connect data intensions and extensions which is the basis of its strength and flexibility. Because all three layers of TC share a common frame-based ontology, the mappings can be maintained internally, giving each container desired agility and autonomy. This provides a solution to the "brittleness problem" mentioned in 3.1.3.1.

3.1.4 Summary of main benefits of the proposed solution

This proposal offers an innovative solution to three important issues in the design of information integration appliance, such as DATALATOR. The offered solution is based on an attempt to address the single most fundamental problem at the core of all three – the intended information interpretation that is the alignment of intentions between producers of information and its consumers. In dynamic organizations, such alignment cannot be assumed. Instead it must be constantly renewed. Moreover, the original interpretation of the information stored in heterogeneous data sources is not always available in *ad hoc* data integration scenarios and may be conflicting between the sources. Correspondingly, the proposed solution is focused on recovery of only that part of the interpretation which is most relevant in a context provided by the consumer when accessing data. The solution employs principles of Emergent Semantics to achieve an incremental and goal-directed process, which sufficiently constrains the space of possibilities.

The main advantage of the proposed solution is that it does not require user to be proficient in any formal logic. User only needs to identify some target relations in local context, from which the system builds a mapping and applies it to the data source. New concepts (topics) are acquired in a form of stereotypes (meta-classes) allowing it to be applied to any data source, regardless of local constraints. Another benefit of the proposed solution is utilization of facets in frame-based ontologies. Facets can be used to further automate the alignment and mapping between different ontologies and data sources, in addition to the reasoning over frames. This can be useful when data sources change after the initial import, and re-mapping of ontology to the data source would typically be required. Instead, facet can be used to "silently" reconfigure the data access without changing the ground ontology – significantly reducing the downtime. Frame facets can also be used to facilitate query translations and data transformations. For example, custom facets can be used to store programs (scripts). In that capacity facets can be very useful for automating syntactic aspects of query translation and data transformation, leaving frames to be dedicated to semantic aspects. Facets can also be used to enable *computed slots* which are the frame slots not originally present in ground ontology, but bound to the original (ground) slots by regular expressions inside custom facets.

In the proposed architecture for DATALATOR users can dynamically create multiple global contexts for information interpretation, based on particular situation. Local contexts are represented by autonomous agents based on TC architecture implemented by OntoBase. Users can directly interact with the agents and identify situations represented in respective local contexts. The global contexts are represented by tuple type systems and populated with data from local contexts. The information can therefore be correctly interpreted based on intended use of data, as well as exchanged and integrated into the global contexts. In essence the users of DATALATOR are in the loop providing the intentional component of strong reference (Figure 18). Alignment of semantic models and formulation of queries is made easier by using more abstract terms of the global contexts with fewer constraints. This reduction of mapping and query effort comes at the price of referring to data indirectly by means of the strong reference.

Based on the proposed architecture DATALATOR can function as a network utility operating at the network protocol level globally enabled by distributed memory architecture like tuple space. Acting as network node DATALATOR can connect to multiple data sources on one end and to multiple semantic type systems comprised of abstract topics (stereotypes) on the other end. Multiple nodes can cooperate using the same type system without exposing their internal semantic models and data structures, which can be important to ensure security and privacy of data. The cooperation is enabled by the global medium comprised o tuple spaces and SONs. Such DATALATOR networks would be highly scalable due to its scale-free architecture (i.e. self-similar at every scale).

3.2 Prototype Architecture

3.2.1 High Level Architecture

The high level DATALATOR architecture is shown in Figure 16. It consists of OntoBase-enabled agents connected to multiple data sources and sharing common tuple spaces. Each agent provides a local context where situation types can be identified by users and mapped to global contexts, stored in meta-classes (topic stereotypes). Global contexts share stereotypes and form global SONs enforcing attachment law based on strong reference semantics. In this architecture new global contexts can emerge in order to allow new agents (acquaintances). Each global context has associated tuple space where data are exchanged between the agents using mapping rules created specifically for that context and implemented by OntoBase as TCs. Each TC has a 3-layered logic organization (Figure 14) with data source's ground ontology and situations' semantic models. Users semi-automatically construct the global contexts mapping situation types to the global topic stereotypes which enforce the attachment law preventing violation of global referential integrity constraints by new acquaintances. In order to do that DATALATOR must perform inferences over tuple type extensions local context.

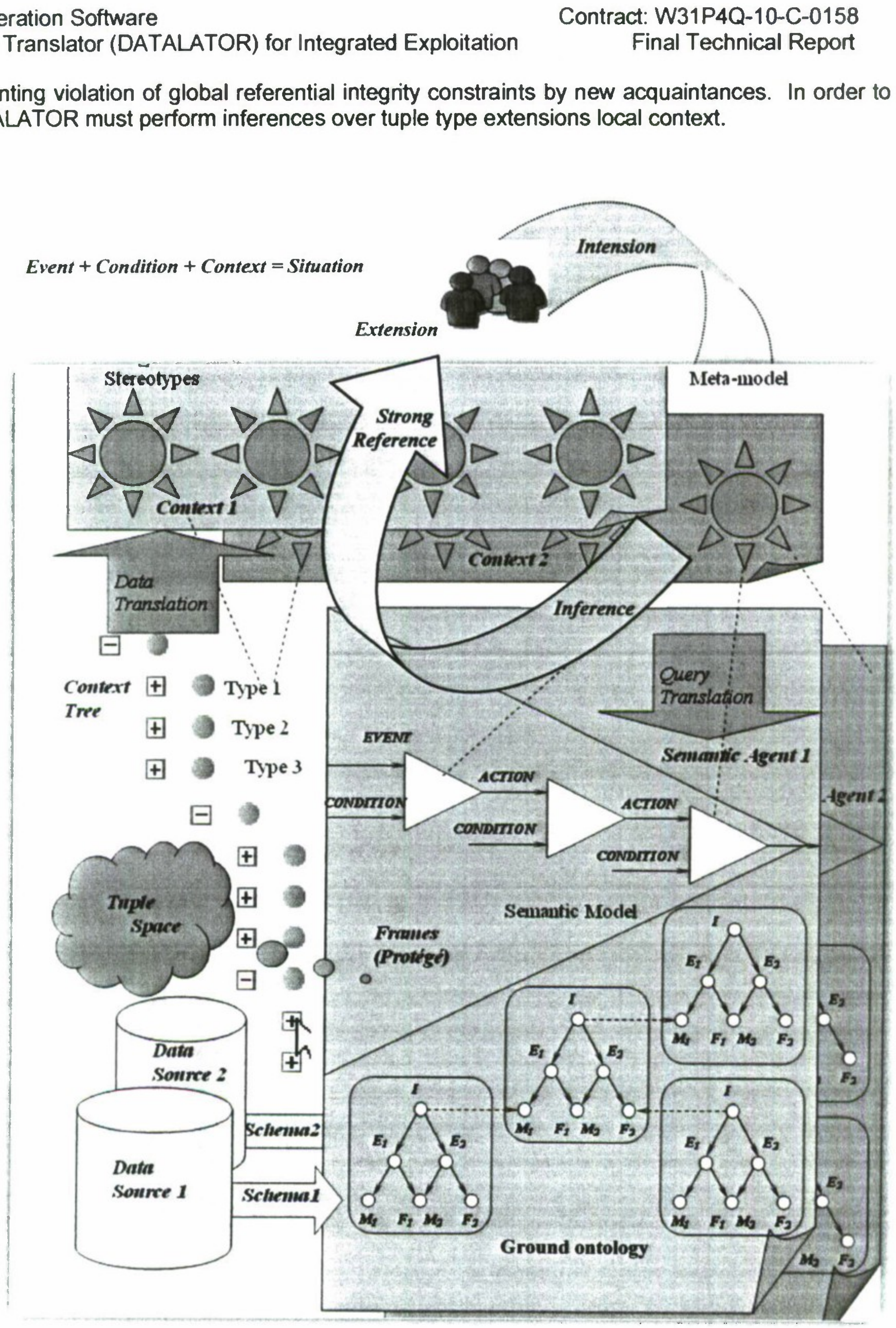


Figure 18: High level DATALATOR architecture

3.2.2 Low Level Architecture

Low level DATALATOR architecture is shown in Figure 17. It consists of Protégé 3.4.4 platform and OntoBase plug-in that connects to multiple data sources and allows users to construct queries using Protégé GUI (Figures 10-15).

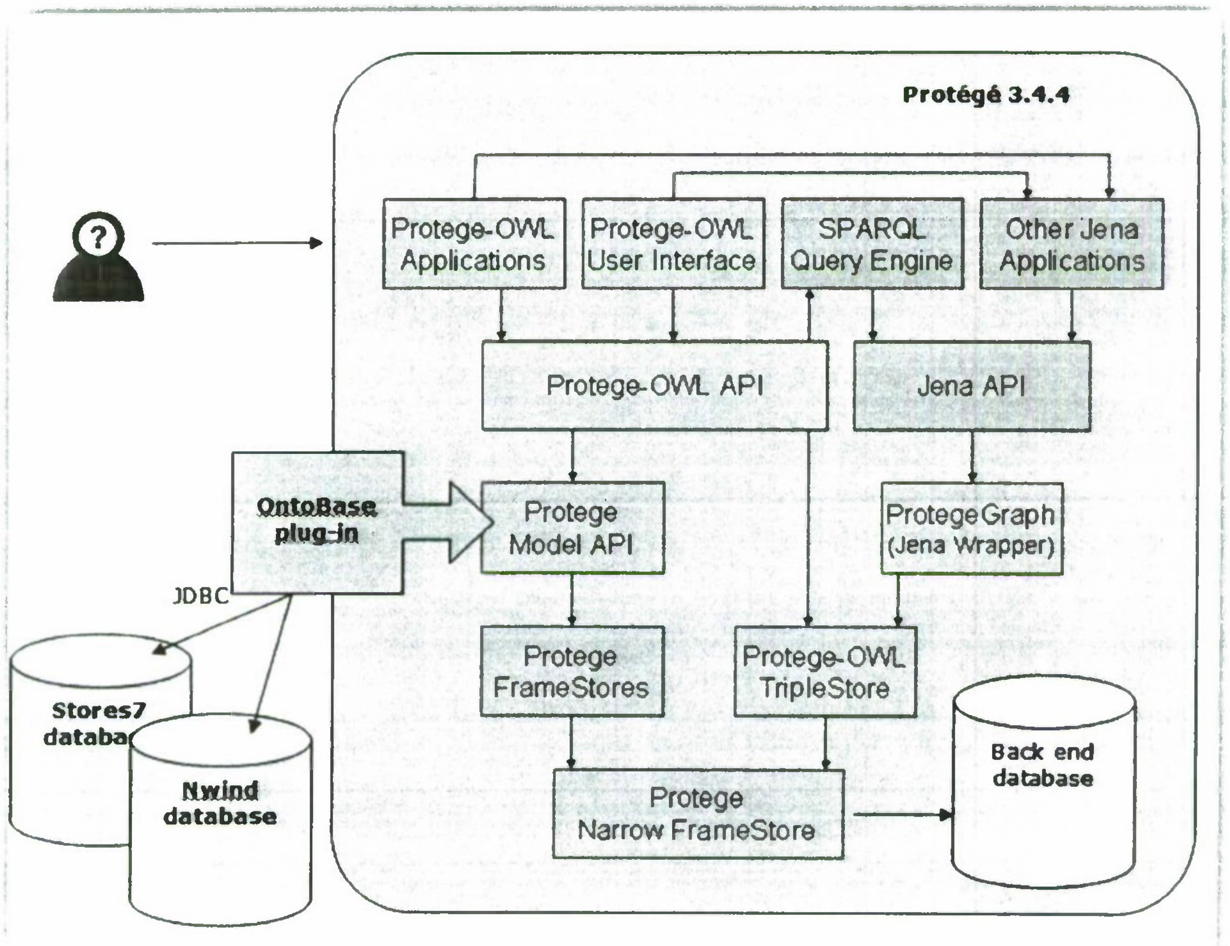


Figure 19: Low level DATALATOR architecture

OntoBase plug-in calls Protégé via its frame API in order to generate class and instance frames as described in 3.1. Protégé stores frames using a back end database which can be an in-memory or external database engine. Protégé plug-in API allows for multiple possible implementations of its back end as well as GUI components. Use of Protégé as platform for DATALATOR prototype was justified by this flexibility to explore various options in phase I and II.

4 PROTOTYPE EVALUATION

During phase I Next Generation software has released OntoBase 3.4.1 which was used to perform the final round of tests

.

4.1 Experiments

The following experiments were designed to empirically evaluate performance of the prototype in meeting the objectives outlined in 1.1 and 2.3. The main focus was placed on quantitatively measuring mapping automation and residual effort required from user. In accordance with the proposed design the effort is broken into three stages: fully automatic ground ontology generation, semi-automatic semantic model discovery and manual posing of queries by user. No attempt was made to quantitatively measure the scope and correctness of answers to the queries, although answers' quality was qualitatively assessed.

4.1.1 Ontology Generation

As explained in 3.1 the ground ontology is automatically generated during the initial import of data source schema into Protégé via OntoBase plug-in.

4.1.1.1 Metrics

Following metrics are applied relative to the size and complexity of each data source

- Time to generate
- Size of ontology
- Number of relations
- Number of ECA chains

4.1.2 Semantic Model Discovery

User-guided model discovery experiments were performed for several data source of various complexity and size.

4.1.2.1 Metrics

Source model discovery effort with respect to the complexity of input meta-data

- Number of relations in schema
- Number of ECA chains
- Number of causal streams
- Time to generated grounding ontology
- Time to generate situation tree
- Depth of situation tree (bounded effort)
- Number of user dialogs (bounded effort)

4.1.3 Question Answering

Questions were posed as Union of Conjunctive Queries (UCQs) over the ontology of situation types discovered in the previous set of experiments.

In order to answer specific questions two types of queries were issued:

a) simple conjunctive query with several sub-goals from the same relation as in $P \wedge Q \wedge R \wedge S$
b) complex join queries:

- Star queries contain a distinct sub-goal that joins with every other sub-goal in a logical star-like formation such as: $P(w, x, y, z) \wedge Q(w, \alpha_1) \wedge R(x, \alpha_2) \wedge S(y, \alpha_3) \wedge T(z, \alpha_4)$
- Chain queries contain sub-goals that join to ones before or after it in a logical chain-like formation as in: $P(\alpha, w) \wedge Q(w, x) \wedge R(x, y) \wedge S(y, z) \wedge T(z, \beta)$

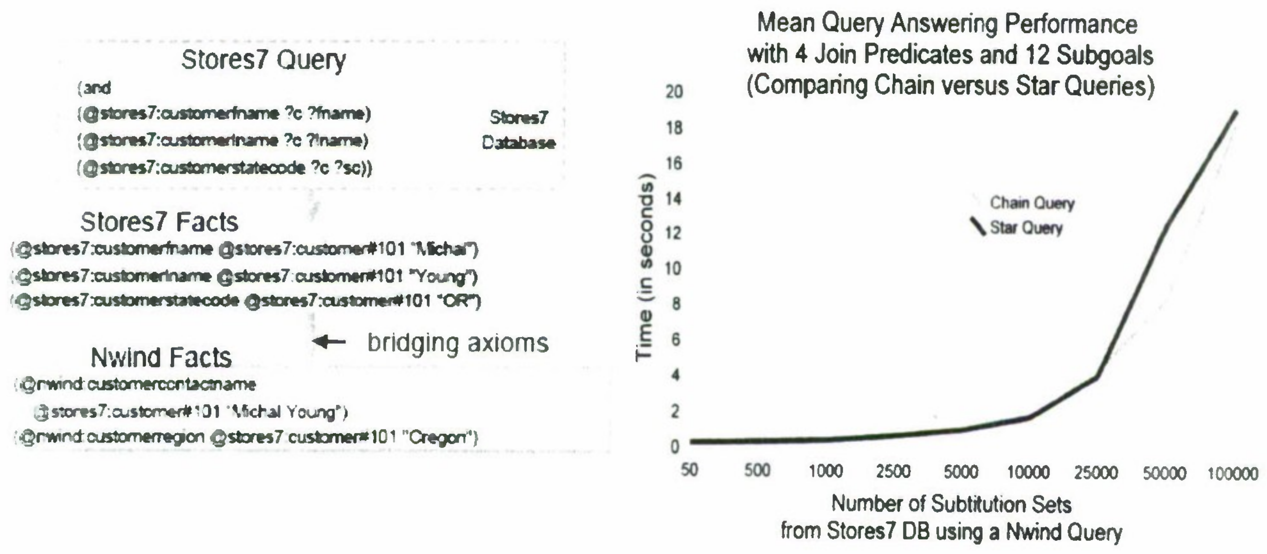
4.1.3.1 Metrics

Answering time was measured relatively to the number of facts fetched from the data source

- Time to process "star" query that joins a single goal with every other sub-goal.
- Time to process "chain" query that join sub-goals to ones before or after it.

4.2 Performance Benchmarks

To adequately assess performance of the prototype it was compared to OntoGrate prototype developed and studied few years ago at the University of Oregon using two of the same data sources Stores7 and Nwind [17]. OntoGrate uses the semantics of the Stores7 ontology to lookup, join and resolve the *statecode* and *statename* bindings based on a *customerregion* from the Nwind ontology. In other words, even though there is no such concept of state code abbreviations in Nwind (Figure 20), it can still resolve this information (Figure 21)



```
(and
(@stores7:customerfname ?c ?fname)
(@stores7:customerlname ?c ?lname)
(@stores7:customerstatecode ?c ?sc))

(@stores7:customerfname @stores7:customer#101 "Michal")
(@stores7:customerlname @stores7:customer#101 "Young")
(@stores7:customerstatecode @stores7:customer#101 "OR")

(@nwind:customercontactname
  @stores7:customer#101 "Michal Young")
(@nwind:customerregion @stores7:customer#101 "Oregon")
```

Figure 20: OntoGrate experiment

4.3 Experimental data

Following data sources were specially selected to meet the requirements set by the proposal D092-007-0197 Section 2.2. The selection spans a large range of size and complexities needed to demonstrate capabilities of the proposed approach.

Name	Size	Description
Joint Consultation Command and Control Information Exchange Data Model (JC3IEDM)	649	Objects are represented with a combination of OBJECT-ITEM and OBJECT-TYPE or their respective subtypes. An OBJECT-TYPE refers to a class object and an OBJECT-ITEM to an individually identified instance. A specific OBJECT-ITEM must be associated with at least one instance of an OBJECT-TYPE. Classifying OBJECT-ITEM as OBJECT-TYPE makes any information that is stored as type data applicable to the item
Mobile Patient Charting System (MPCS)	131	*Patients, Doctors, Nurses* and *Institutions* all have many-to-many relationship between them, such as *Admission, Assessment* and *Intervention.* Instances of the relationships are all attached to medical services via temporal classes, such as *Certification Period* and *Visit*. Creation of the instances is further constrained by many-to-one relationships to a fixed nomenclature of service types and rules.
Sakila, movie rental store sample MySQL database	16	Classes *Film* and *Actor* have a many-to-many relationship between them, i.e., a movie can have many actors in it and an actor can act in multiple movies. Classes *Inventory* and *Store* have a one-to-many relationship between them, i.e., a store can have multiple copies of a movie (inventory) but an inventory item can belong to only one store.
Stores7, fictitious sporting-goods distributor	10	This database includes such tables as *Customer, Orders, Items, Stock, Catalog* etc. The *Customer* table contains information about the retail stores that place orders from the distributor. An *Order* can include one or more *Item(s)*. More than one *Manufacturer* can supply an *Item*.
Nwind, sample database from Microsoft	8	Very similar to the above, but using *Supplier* and *Shipper* instead of *Manufacturer*. There are also some subtle differences in representing *Customer* information.

4.4 Experiment Results

Following experimental results were obtained from running the prototype on Intel Core i7 PC with 6GB of RAM and Windows 7 OS.

4.4.1 Ontology Generation

Data source from 4.3 were loaded in a dedicated MySQL database engine and then imported into Protégé using OntoBase plug-in

Data source	Tables	Generate time (sec.)	Ontology relations	Ontology size (KB)	ECA chains
JC3IEDM	649	998.9	9726	137614	1596
MPCS	131	201	1761	29401	245
Sakila	16	17.3	23	876	12
Stores7	10	5.24	3	302	3
Nwind	8	5.35	5	331	5

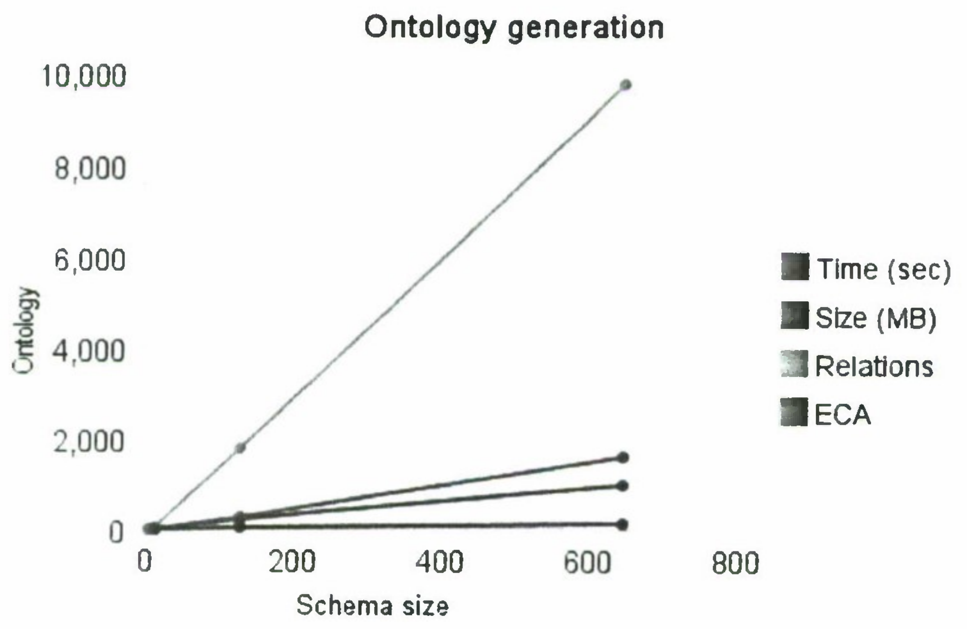


Figure 21: Ontology generation time and size

4.4.1.1 Analysis

The results above show linear complexity of used algorithms in generation time and space relative to size of database schema. The absolute time and space is within reasonable range of up to 15 min and 137MB which should be acceptable source initialization for most applications.

4.4.2 Source Model Discovery

The following are the results of experiments conducted according to D092-007-0197 Section 2.2.3a.

Data Source (Decision tree)	**Generate** (sec.)	**Number of chains** (tree width)	**Length of chains** (tree depth)	**Effort** (interactions)
JC3IEDM				
Object-Item-Status	16.333	25	3	9
Road	14.243	14	3	13
Person	13.728	4	1	5
Context	13.338	6	2	7
Objective	14.43	8	2	9
MPCS				
Patient-Intervention	1.467	35	5	15
Patient-Medication	0.562	15	3	11
Care-Plan	0.484	11	2	5
Diagnosis	0.468	9	2	9
Sakila				
Store Movies	0.172	9	4	7
Movie Rentals	0.281	9	4	5
Customer Rentals	0.141	8	3	5
Stores7				
Customers	0.078	1	4	5

4.4.2.1 Analysis

The results clearly demonstrate bounded effort and time spent by untrained user without prior knowledge of the data source internal structure.

Using situation trees is a main feature of our approach that allows any user to choose among multiple alternative flows of information, as most relevant in a given situation. This simple interaction replaces what would be a complex task of analyzing the data source structure and finding ways to retrieve needed information from it. Moreover, a product of such complex analysis may or may not be useful in other situations. And in either case it cannot be used directly to access and manipulate the data, but only as a basis for applying data retrieval tools and user interfaces (eg. SQL). Our prototype, on the other hand, guides the user through bounded number of simple and intuitive steps directly to retrieval of data without using any other tools. At the same time our system is recording the user choices and organizes all the results in a reusable and standard form of ontology.

Our experiments used data sources from wide range of domains, sizes and structural complexity. However, the resulting number of required user interaction has a much narrower range of 5-15. The complexity of interactions as measured by width and depth of the situation tree also tend to stay within a reasonable range. This leads us to a preliminary conclusion that our prototype achieves desired DATALATOR characteristics of bounded effort and time in a wide range of data sources.

4.4.3 Question answering

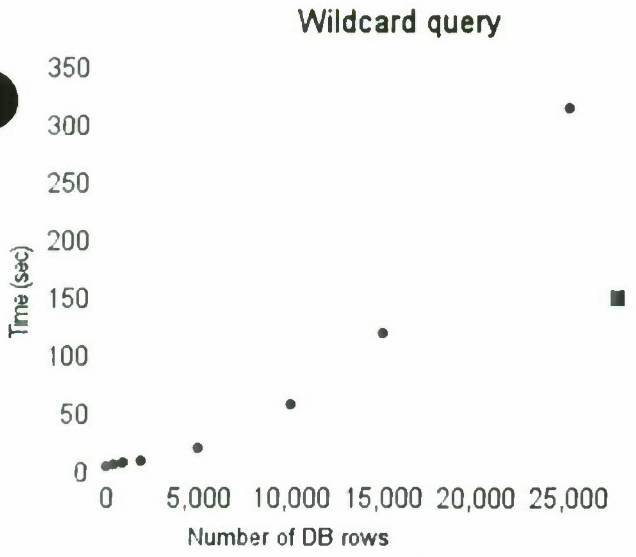
Question answering experiments were performed using MySQL database running on a dedicated server. In order to calculate the offset related to the difference in settings a simple question that does not require any semantic translation – was processed first:

Who are all the Customers in the given city?

Number of DB rows		**500**	**1,000**	**2,500**	**4,000**	**5,000**	**10,000**	**20,000**	**25,000**	**35,000**	**50,000**
Query Answer (sec.)	**OntoGrate**	**1.5**	**1.5**	**2**	**3**	**3**	**9**	**41**	**66**	**145**	**312**
	OntoBase	**1.5**			**7.6**		**21**		**103.5**		**322.4**

Table 1: OntoBase performance measured relative to OntoGrate benchmark

After the answers were loaded into Protégé as individual instances of *Customer* class, the project was re-loaded and "atomic" queries automatically issued for each instance in the ontology to bind with the original database record. This was done to offset the impact of Protégé API on loading of data. In case of the new wildcard queries Protégé instance frames are created by calling frame API as new rows are read from database. However, in case of re-loading previously created and saved Protégé instances the bulk loading of instances from XML is done first, before atomic queries are issued and pre-loaded frames are populated with the data.



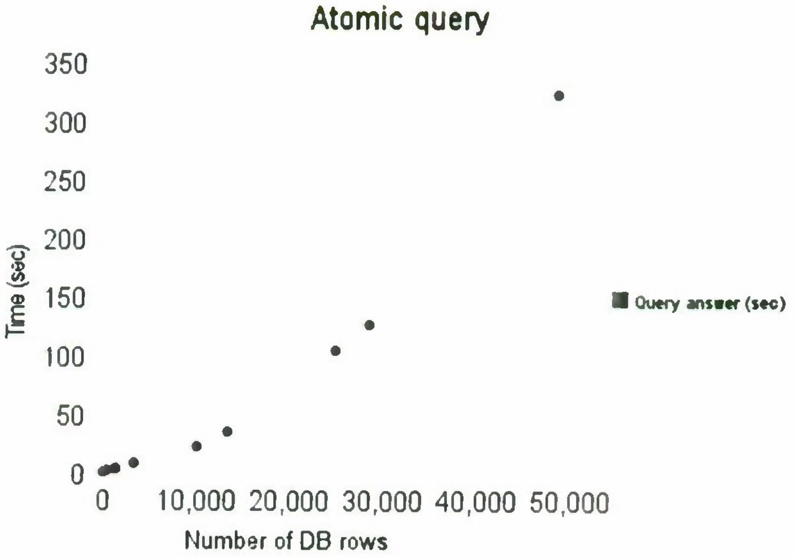


Figure 22: Comparing performance of wildcard query vs. atomic query

4.4.3.1 Analysis

This impact of the Protégé API is visible as a 2x performance difference in combined query answering and data loading time between wildcard and atomic queries retrieving exactly the same data. In both cases the time was exponential to the number of rows loaded, just as it was in the benchmark OntoGrate experiments (Figure 20). Additional re-runs of the experiments were also done with various

JVM flags to isolate the impact of Java memory allocation and garbage collection algorithms. No significant impact on performance was observed from that end.

Finally, star and chain queries were tested and their performance measured according to 4.1.3.1. There was no measurable difference between the two cases. The combined time of query translation and data transformation was exponential to the number of rows retrieved (Figure 23).

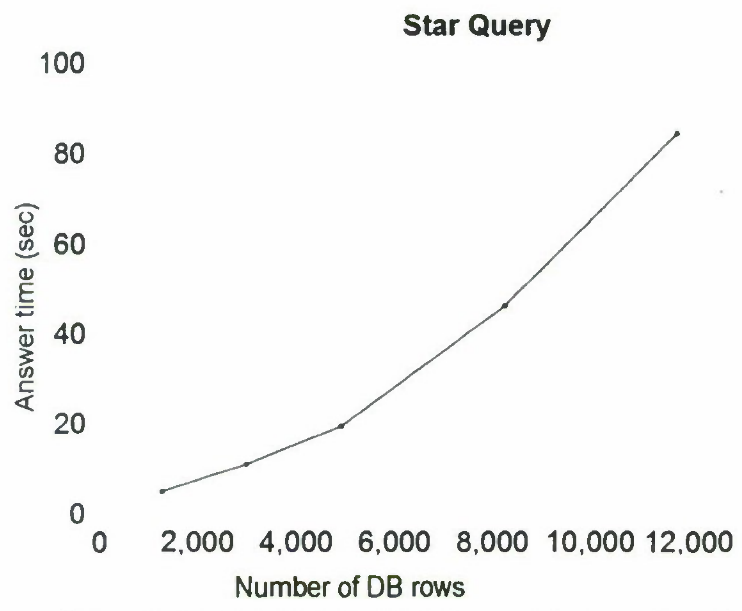


Figure 23: Combined query translation and data transformation

4.4.4 Additional Observations

In addition to quantitative performance measurement reported above the quality and scope of answers were also observed. To that end a set of questions was developed (APPENDIX A) and posed using the prototype. The answers obtained from different data sources are not reported due to limited space, but the SQL queries automatically generated by prototype are listed for every question. The answers loaded into Protégé frame instances were selectively checked and compared to the results obtained by directly executing the SQL. In all cases the data was accurately and completely represented in frames, with some platform-specific data types correctly converted to Protégé frame data types.

Although outside the scope of this project, making changes to data was also tested and worked well for simple data types like String and Integers. This additional capability of the prototype will be exploited in future commercial product enabling round-trip data exchange with any connected data source.

5 FUTURE PLANS

5.1 Phase I as Foundation for Phase II

Phase I captured the requirements and developed a preliminary design of a prototype to be integrated tested and demonstrated in Phase II. The goal of Phase I was to prove the feasibility of the technology and advance development towards a real world demonstration. The results of testing demonstrated that the technology did, in fact, do what it was intended to do. Phase I was successful in validating the technology. Phase I results will now enable the shift from a primary validation of concept to building of a TRL5 prototype, and specification and design of the necessary enhancements to application and systems architectures to deliver a real world demonstration by the end of Phase II.

A key consideration of any real world demonstration is not only that the prototype can demonstrate its intended capability, but also perform sufficiently to deliver that capability within realistic parameters. The empirical evaluation of TRL4 prototype delivered in Phase I showed clearly its limited scalability in volume of data that it can process. Testing identified that the most significant impact on the prototype performance came from the Protégé ontology engine. Resolving this issue is a clear requirement of Phase II, and it is believed that when it is resolved, a linear performance curve will be achieved.

5.2 Phase II as Foundation for Commercial Products

During Phase II, Next Generation Software will leverage the insights gained during Phase I to produce a TRL5 prototype, initially mitigating performance issues within the current technology platform and validating the ability of the platform to scale to a commercial application. The risk remains, however, that the identified performance issues are inherent to the current technology platform. To manage this risk, Next Generation Software will also identify commercially available platforms and middleware (computer software that connects software components) that support the application requirements and show strong potential to resolve the performance issues. The evaluation of alternative platforms and middleware will consider: (i) technical design and performance, (ii) the ability of the company providing the platform and/or middleware to provide commercialization assistance, and (iii) the ability of the company to provide development resources to Phase II. The company has already secured a Letter of Intent from a potential partner, SOA Trader, Ltd.(http://www.soatrader.com). SOA Trader, Ltd. was founded in 2006 with the objective to accelerate companies' IT outsourcing efforts through the promotion of XML services as reusable and managed software components, which are available on-demand. SOA Trader's Letter of Intent is included in Appendix B.

During this phase, Next Generation Software will expand relationships with commercial software and systems vendors in an effort to identify a partner or partners with strong commercial experience in the design and testing of commercial software products. Working with these partners, a Phase III product road map will be developed that identifies: (i) the application architecture, system architectures, and middleware necessary to deliver the product end-to-end in a commercial environment, (ii) identifies a realistic time line to mature the prototype toward that goal, and (iii) captures the financial investments required to achieve it. As part of that effort, the partners would provide guidance and facilities to perform real world, commercial scale testing of the product, and assist with resolution of any issues that rigorous testing identifies.

To assist Next Generation Software achieve the goals of Phase II, an Advisory Board will be formed that will be comprised of: (i) serial entrepreneurs who have successfully commercialized software companies in the past, (ii) senior leaders from the alternative investment community who t have successfully funded early-stage companies, and (iii) industry experts with specialized knowledge of key vertical markets. The company will also expand its management team with key individuals who have a track record of commercializing software products and building the operational capabilities to support them. These key managers will formalize the commercial structure of the business, identify and define operational requirements, procedures and controls, identify funding requirements and develop financial models and reports, help define a business development strategy, and assist with efforts to secure Phase II funding. All of these activities will enable deployment of the product into commercial environments in Phase III. The company has already identified a senior-level management executive, Gian Zoppo, who has successfully commercialized several application service providers in healthcare information and financial fraud prevention. Mr. Zoppo's letter of intent is included in the Appendix B.

5.3 Initial Phase II Prototype Specification & Design

Mitigation of the indentified performance issues will initially drive prototype specification and design of the TRL5 prototype. Beyond the immediate performance issues, the build out of a TRL5 prototype will increase the maturity of the system architecture and middleware to produce a real world example of both the application and the necessary systems to deliver Ontobase end-to-end into a commercial environment. Next Generation Software will leverage Advisory Board and partner expertise to identify potential risk areas associated with the Phase III product road map, including technology architecture, middleware, enabling technologies, the pace of technology change, and other potential impact to prospective application areas.

Another focus will be the specification and design of a security management module that wraps the application in a secure cocoon away from prying eyes, and gives an administrator the ability to segregate users and data on a need to know basis. Enabling the broadest possible data set to contribute maximum inputs is only possible within a managed security environment that would allow access to secure systems without the risk of compromising sensitive data. During Phase II, Next Generation Software will design and build a robust security architecture, again seeking out the appropriate partner or partners with deep domain knowledge and commercial presence who could provide both guidance and real world testing capabilities.

In addition to the larger architectural considerations of the product, Phase II prototype development will deliver user interface (UI) and error handling enhancements, The UI is the connection between the user and the application, and it is critical to guide the user with an intuitive presentation of application elements, and support them with robust and clear error handling. Next Generation Software will expand both of these requirements during Phase II prototype development. Across all development and commercialization activities, formal change management and risk management processes and systems will be developed, documented, and incorporated into all activities to ensure accuracy of information, and tight controls leading up to the production of a TRL5 prototype. The company will also enhance the document set for TRL5 to facilitate partnering, user testing, and training.

5.4 Efforts in support of Phase III Transition

Next Generation Software's commercialization strategy is based on dual introduction of (i) free open source OntoBase plugin for major Integrated Development Environments (IDE), and (ii) later rollout of "Situational Application" Platform-as-a-Service (PaaS) solution delivered via the "Cloud." (codename "OntoSpace"). The free open source version will enable small scale rapid integration of "data mashup" applications, driving opportunities for strategic partnering with IDE providers, revenue sharing with their clients or direct revenue for enhancements requested by IDE providers. OntoSpace situational applications delivered over the Cloud will generate recurring "subscription" revenue on a model similar to other PaaS providers such as Salesforce.com. An additional source of revenues is expected to be licensing of the original OntoSpace Intellectual Property (IP), and investment by Primary vendors interested in integration of OntoSpace into their products or acquisition of the company.

Software to establish and maintain a situational awareness in dynamic data-intensive open environments continues to gain attention in the market. Situational applications are built on-the-fly to solve a specific business problem, which fits neatly with the agile development approach that is fast becoming the preferred development approach within the Fortune 1000 companies. Next Generation Software believes that there is an unlimited potential market for the OntoSpace product, however, initial business development activity will focus on the Federal Software Products Market, because demand for vendor-furnished software products by the U.S. government is expected to increase from $6.5 billion in 2010 to $8.4 billion in 2015 at a compound annual growth rate (CAGR) of 5.2 percent according to a recent report by INPUT. The company will be well positioned to take advantage of this growth. OntoSpace delivered over the Cloud would provide demonstrable value and utility, and a compelling case could be made for a Platform as a Service offering. Focus on the government market will also include state and local governments, specifically around first responders and law enforcement agencies, though there is no intrinsic limitation, and any agency could benefit from our innovative approach. Within the private sector, we believe that there are opportunities to develop market share in financial services and healthcare and we will pursue these markets as well.

The gap that must be bridged between Phase II and Phase III revolves largely about the business and marketing activities necessary to support the product. Business commercialization activities include: (i) structural and legal definition of the company's business model and product and services lines, (ii) implementation and operational staffing levels, and (iii) any necessary licensing and certifications that may be required. These activities must be performed in Phase II to pave the way for a smooth transition to Phase III. A well-defined strategy to scale the business is just as important. An early-stage company cannot assume that it will just grow as needed. A primary and secondary strategy is needed that involves both growing the company and leveraging partner relationships to meet growth requirements. Just as important as having the mechanisms to scale business is having the right offering to bring to the marketplace. To do this, a deep understanding of both the packaging and pricing of current commercial products and service arrangements is necessary.

A key activity in support of these requirements will be identifying and winning the first Beta customers. Next Generation Software will leverage the Advisory Board and partner relationships to identify and win Beta Customers during Phase II. These anchor customers will provide critical input into the packing of the offering, the commercialization of the product, and its support and operational requirements. They will be able to identify specific business requirements for the product, identify competitor solution shortfalls, validate the regulatory and pricing models, and help identify future product enhancements. They can also serve as reference customers supporting additional marketing activities. Beta customers also provide revenue into the company that is critical to securing additional funding for Phase III.

In addition to SBIR and other grants, additional funding will be necessary for Phase III transition. Next Generation Software will pursue private funding both from partners, and from private investors. Next Generation Software expects to obtain of $750K-$950K in matching private investments from angel investors for SBIR Phase II in 2011 and 2012, for combined "revenue" of $1.5M - $1.9M from investment through 2012 which includes $200K in Phase II Enhancement. The company intends to leverage the Transition Support Pilot Program to identify additional funding sources. The company will pursue additional investments of $3M from angel investors and/or venture capital to advance Ontobase to TRL7. A summary of the Key Milestones and the funding tied to each can be seen in the chart below.

5.4.1 Key milestones

Milestone	Phase I 2010	Phase I 2010	Phase II 2011	Phase II 2012	Phase III 2013	Phase III 2014
Technology	TLR 3	TRL4	TRL5	TRL 5	TRL 6	TRL 7
Revenues					$250K	$750K
SBIR	$50K	$50K	$375K	$575K	TBD	TBD
Investments			$375K	$575K	$1M	$2M

REFERENCES

1. **Dittrich, Patrick Ziegler and Klaus R.** Data Integration — Problems, Approaches, and Perspectives. Andreas L. Opdahl, and Sjaak Brinkkemper John Krogstie. *Conceptual Modelling in Information Systems Engineering*. 2007, Berlin : Springer, pp. 39–58.

2. **Arnon Rosenthal, Len Seligman.** *Pragmatics and Open Problems for Inter-schema Constraint Theory.* 2006. Proceedings of the 22nd International Conference on Data Engineering Workshops. p. 1.

3. **Jie Bao, Jiao Tao, Deborah L. McGuinness, Paul R. Smart.** *Context Representation for the Semantic Web.* Raleigh, NC US : s.n., 2010. WebSci10: Extending the Frontiers of Society On-Line.

4. **Cristiana Bolchini, Carlo A. Curino, Elisa Quintarelli, Fabio A. Schreiber, Letizia Tanca.** A data-oriented survey of context models. *SIGMOD Record.* December 2007, Vol. 36, 4, pp. 19-26.

5. **Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, Daniele Riboni.** A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing.* April 2010, Vol. 6, 2, pp. 161-180.

6. **Yabloko, Len.** *Logic Stratification and Alignment for Heterogeneous Distributed Information Systems.* Banf, Canada : 7th International Conference on Artificial Intelligence and Soft Computing, 2003.

7. **A. Doan and A. Y. Halevy,** *Semantic-integration research in the database: A brief survey.*1, March 2005, AI Magazine, Vol. 26, pp. 83–94.

8. **Halevy, A.Y., et al.** *Schema Mediation in Peer Data Management Systems.* 2003. 19th International Conference on Data Engineering .

9. **Diego Calvanese, Evgeny Kharlamov, Werner Nutt, Camilo Thorne.** *Aggregate Queries over Ontologies.* Napa Valley, California, USA : s.n., 2008. ONISW'08.

10. **Libkin, Leonid and Sirangelo, Cristina.** *Data Exchange and Schema Mappings in Open and Closed Worlds.* Vancouver, BC, Canada : s.n., 2008. PODS'08.

11. **Tzy-Hey Chang, Edward Sciore.** A universal relation data model with semantic abstractions. *IEEE Transactions on Knowledge and Data Engineering.* 1992, Vol. 1, 4, pp. 23-33.

12. **Goguen, Joseph A.** Data, Schema, Ontology and Logic Integration. *Logic Journal.* November 2005, Vol. 6, 13, pp. 685-715.

13. **Yabloko, Len.** *Software Architecture with Emergent Semantics.* Miami, USA : s.n., 2005. 7th International Conference on Enterprise Information Systems.

14. **Partridge, Chris.** *Business Objects: Re-Engineering for Re-Use.* s.l. : Butterworth-Heinemann Ltd, 1996.

15. **Shen, W., DeRose, P., McCann, R., Doan, A., and Ramakrishnan, R.** *Toward best-effort information extraction.* Vancouver, Canada : s.n., 2008. SIGMOD International conference on Management of data. pp. 1031-1042.

16. **Anish Das Sarma, Xin Dong , Alon Halevy.** *Bootstrapping pay-as-you-go data integration systems.* Vancouver, Canada : s.n., 2008. ACM SIGMOD international conference on Management of data. pp. 861-874 .

17. **Dou, Dejing and Paea, Paea LePendu.** *Ontology-based Integration for Relational Databases.* 2005. International Conference on Ontologies, Databases and Application of Semantics (ODBASE).

18. **Karl Aberer, Tiziana Catarci, Philippe Cudr´e-Mauroux, Tharam Dillon, Stephan Grimm, Mohand-Said Hacid, Arantza Illarramendi, Mustafa.** Emergent Semantics Systems. *Lecture Notes in Computer Science.* 2004, Vol. 3226, pp. 14-43.

19. **Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, Ilya Zaihrayeu.** *Data Management for Peer-to-Peer Computing: A Vision.* Madison : s.n., 2002. Int'l Workshop on the Web and Databases (WebDB).

20. **Zhang Zili, Zhang Chengqi.** Agent-Based Hybrid Intelligent Systems. *Lecture Notes in Computer Science.* 2004, Vol. 2938 .

21. **Ahuja, S., N. Carriero and D. Gelernte,.** Linda and friends. *IEEE Computer.* August 1986, pp. 26-32.

APPENDIX A

Sakila

"What movies are available in a given store?"

In this use case a user plays role of imaginary store staff who logs-in to check available movies on behalf of imaginary customer. The user then goes through the list while checking availability of each movie. If the store inventory has at least one copy with not empty return date, then the movie is available:

```
SELECT staff.store_id, staff.password , address.phone FROM sakila.store
JOIN sakila.staff ON store.manager_staff_id=staff.staff_id
JOIN sakila.address ON store.address_id=address.address_id
WHERE staff.username='[login]'

SELECT film.title, inventory.inventory_id FROM sakila.store
JOIN sakila.inventory ON store.store_id=inventory.store_id
JOIN sakila.film ON inventory.film_id=film.film_id
WHERE store.store_id=[store_id]

SELECT rental.return_date FROM sakila.inventory
JOIN sakila.rental ON inventory.inventory_id=rental.inventory_id
JOIN sakila.film ON inventory.film_id=film.film_id
JOIN sakila.customer ON rental.customer_id=customer.customer_id
WHERE inventory.inventory_id= [inventory_id]
```

"Who had rented a given movie?"

In this use case a user will find a movie by its title and list all instances of renting the movie by any customer, regardless from which store:

```
SELECT customer.last_name, rental.rental_date FROM sakila.inventory
JOIN sakila.rental ON inventory.inventory_id=rental.inventory_id
JOIN sakila.film ON inventory.film_id=film.film_id
JOIN sakila.customer ON rental.customer_id=customer.customer_id
WHERE film.title='[title]'
```

"Which movies had given customer rented?"

In this use case a user will find a customer by last name and list all instances of renting a movie by the customer, together with a store address:

```
SELECT store.store_id, film.title , rental.rental_date, address.address FROM sakila.inventory
JOIN sakila.rental ON inventory.inventory_id=rental.inventory_id
JOIN sakila.film ON inventory.film_id=film.film_id
JOIN sakila.store ON inventory.store_id=store.store_id
JOIN sakila.customer ON rental.customer_id=customer.customer_id
WHERE customer.last_name='[name]'
```

"What is a customer payment history?"

In this use case a user will find a customer by last name and list all payments made by the customer:

```
SELECT count(*) FROM sakila.payment
JOIN sakila.customer ON payment.customer_id=customer.customer_id
WHERE customer.last_name='[name]'
```

MPCS

"What is a patient diagnosis?"

In this use case a user will find a patient by last name along with all his/her diagnosis codes:

```
SELECT diag_ref.emcdiag_code1, diag_ref.description FROM mpcs.diag_ref
JOIN mpcs.patientdiagnosis ON diag_ref.diag_ref_id=patientdiagnosis.diag_ref_id
JOIN mpcs.intake ON patientdiagnosis.intakeid=intake.intakeid
JOIN mpcs.patient ON intake.patientid=patient.patientid
WHERE patient.lastname=[name]
```

"When should a patient expect visits by nurse?"

In this use case a user will find a patient by last name along with all scheduled visits to this patient:

```
SELECT employee_key.emfirstname visit.starttime visit.visitdate FROM mpcs.admission
JOIN mpcs.visit ON admission.admissionid=visit.admissionid
JOIN mpcs.intake ON admission.admissionid=intake.intakeid
JOIN mpcs.patient ON intake.patientid=patient.patientid
JOIN mpcs.employee_key ON intake.employeeid=employee_key.employeeid
WHERE patient.lastname=[name]
```

"What medical interventions had been done for a patient?"

In this use case a user will first find all visits made to the patient and then list all interventions made during that visit:

```
SELECT visit.visitid FROM mpcs.visit
JOIN mpcs.admission ON visit.admissionid=admission.admissionid
JOIN mpcs.intake ON admission.admissionid=intake.intakeid
JOIN mpcs.patient ON intake.patientid=patient.patientid
WHERE patient.lastname='[name]'

SELECT patientintervention.intervdescription FROM mpcs.visit
JOIN mpcs.patientintervention ON visit.visitid=patientintervention.visitid
WHERE visit.visitid=[visitid]
```

"What medications is patient taking?"

In this use case a user will find a patient by last name along with all medications the patient is taking and the doctor who prescribed the medication:

```
SELECT patientmedication.medname FROM mpcs.patientmedication
JOIN mpcs.doctorpatient on patientmedication.doctorpatientid=doctorpatient.doctorpatientid
JOIN mpcs.intake ON doctorpatient.intakeid=intake.intakeid
JOIN mpcs.patient ON intake.patientid=patient.patientid
WHERE patient.lastname='[name]'
```

Stores7

"Who are all the Customers in a given city?"

This use case will be used to measure baseline performance for further experiments of query translation and data transformation.

```
SELECT customer.customerfname , customer.customerlname FROM stores7.customer
WHERE customer.customercity='[city]'
```

"How many orders came from customers in a given city?"

```
SELECT count(*) FROM stores7.order
JOIN stores7.customer ON order.ordercustomernumber=customer.customernumber
WHERE customer.customercity = '[city]'
```

"What products from a given manufacturer were sold in a given city?"

```
SELECT stock.stockdescription FROM stores7.item
JOIN stores7.stock ON item.itemstocknumber=stock.stocknumber
JOIN stores7.order ON item.itemordernumber=order.ordernumber
JOIN stores7.customer ON order.ordercustomernumber=customer.customernumber
JOIN stores7.manufacturer ON item.itemmanufacturercode=manufacturer.manufacturercode
WHERE manufacturer.manufacturername='[name]' AND customer.customercity='[city]'
```

Nwind

"How many units of a given product where sold in a given region?"

```
SELECT count(*) FROM nwind.orderdetail
JOIN nwind.product ON orderdetail.orderdetailproductid=product.productid
JOIN nwind.order ON orderdetail.orderdetailorderid=order.orderid
JOIN nwind.customer ON order.ordercustomerid=customer.customerid
WHERE product.productname='[product]' AND customer.customerregion='[region]'
```

APPENDIX B

OÜ SOA Trader - Letter of Intent

Peep Kungas
CEO
OÜ SOA Trader
Akadeemia tee 21
12618 Tallinn
Estonia

Date (10/14/2010)

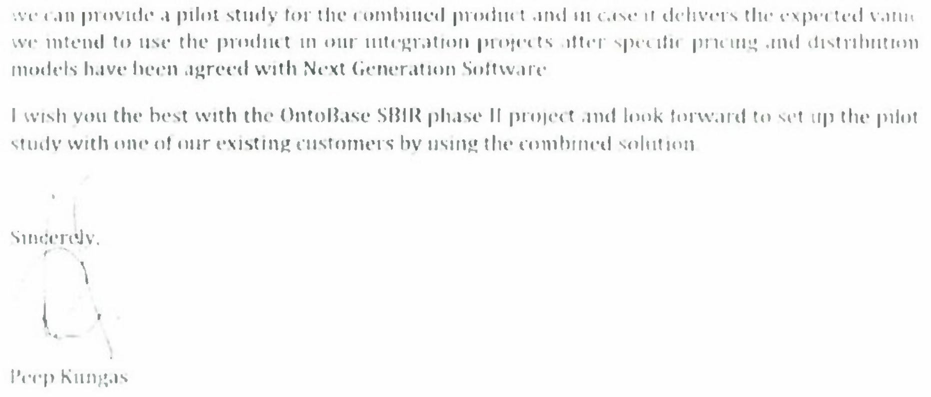
Dear Len Yabloko,

SOA Trader is pleased to announce its intention to collaborate with Next Generation Software on a potential SBIR Phase II contract "Database Translator (DATALATOR) for Integrated Exploitation". If brought to fruition, SOA Trader would enter into a cross-licensing with Next Generation Software to enable the exchange intellectual property.

SOA Trader has products and experience gained through its contracts and employees in the areas of semantic enrichment and large-scale analysis of data models for governmental informations systems. Furthermore, SOA Trader is developing currently a product, Data Mapping Accelerator (DMA), which would first simplify construction of common data models (technically implemented as OWL ontologies) from large collections of database and XML schemas and then linking them with specific data models. Through ongoing collaboration with Progress Software Corporation we are exploring the options to integrate this tool into their DXSI product portfolio in order to simplify creation of mappings between data models and software component descriptions, which finally will lead to higher usability of the tool and increases productivity of its users.

The foundational idea and specific algorithms of DMA have been validated on an analysis of the Estonian State Information System, which is a federation of hundreds of domain-specific information systems using altogether thousands of database tables. The empirical results show that the product would speed up common data model construction for a federated information system at least 5 times in terms of project length and man-hours required to perform this task manually.

By analyzing our preceding e-mail discussions during the past months I have reached an understanding that there is a great opportunity for synergy, which can be achieved by complementing OntoBase and DMA with each other. While OntoBase simplifies aggregation of data from data sources, which are linked together through ontologies, DMA simplifies creation of those ontologies and linking data source descriptions with the ontologies. After discussing usage of the combined product with my local business partners we came to the conclusion that



we can provide a pilot study for the combined product and in case it delivers the expected value we intend to use the product in our integration projects after specific pricing and distribution models have been agreed with Next Generation Software

I wish you the best with the OntoBase SBIR phase II project and look forward to set up the pilot study with one of our existing customers by using the combined solution

Sincerely,

[illegible]

Peep Kungas

Gian Zoppo
& Associates

Gian R. Zoppo
278 Harwood Avenue
Sleepy Hollow, NY 10591

October 30, 2010

Len Yabloko
Next Generation Software
52 Strawtown Road
New City, NY 10956

Letter of Intention to Join Next Generation Software Management Team

Dear Len;

I am delighted to confirm my intention to join the Next Generation Software management team as Chief Executive Officer (CEO), leading the commercialization of the Ontobase product(s) upon award of SBIR Phase II. As we have discussed, I have a track record of successful commercialization of venture capital backed start up and early stage companies, as well as 10 years of executive management experience spanning a broad set of business, financial, and technical disciplines within enterprises up to $6 billion in revenues, and operations spanning 92 offices in 14 countries.

During my most recent engagement, I was retained to commercialize a software as a service business unit of Fortent, that was a strategic lynch-pin of a $200 million dollar investment by Warburg Pincus in the risk and financial crime solutions market. The unit delivered enterprise-scale software applications to detect money-laundering and fraud related to terrorist funding as a service to leading banks. The successful effort was identified as key to Fortent's acquisition by Actimize in 2009, consolidating over 200 clients worldwide and including all top 10 global and top 10 U.S. banks.

Prior to Fortent, I was retained by a venture capital investment group to take a paper concept and turn it into a commercial business, directing all aspects of establishing commercial operations for the application service provider delivering electronic medical records to medical practices and hospitals. The successful effort captured some of the leading medical practices within the New York Metro area and forged key strategic partnerships with industry leaders such as Thomson Medical.

Additionally, during my previous seven years on executive teams within the Omnicom Group, I launched numerous new businesses delivering global client facing and enterprise-scale software applications via the Internet to Fortune 1000 clients including Mercedes Benz, Procter & Gamble, and HP.

I look forward to hearing of your successful proposal and working with you to build a successful business.

Sincerely,

[illegible]